(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 December 2000 (21.12.2000)

PCT

(10) International Publication Number
WO 00/77592 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number: PCT/US00/15860

(22) International Filing Date: 9 June 2000 (09.06.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/139,071	11 June 1999 (11.06.1999)	US
09/345,215	30 June 1999 (30.06.1999)	US
60/144,693	20 July 1999 (20.07.1999)	US
60/149,276	17 August 1999 (17.08.1999)	US

(71) Applicant *(for all designated States except US)*: THE FOXBORO COMPANY [US/US]; 33 Commercial Street, Foxboro, MA 02035 (US).

(72) Inventor: JOHNSON, Alexander; 2203 Stoney Brook, Houston, TX 77063 (US).

(74) Agents: POWSNER, David, J. et al.; Nutter, McClennen & Fish, LLP, One International Place, Boston, MA 02110-2699 (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

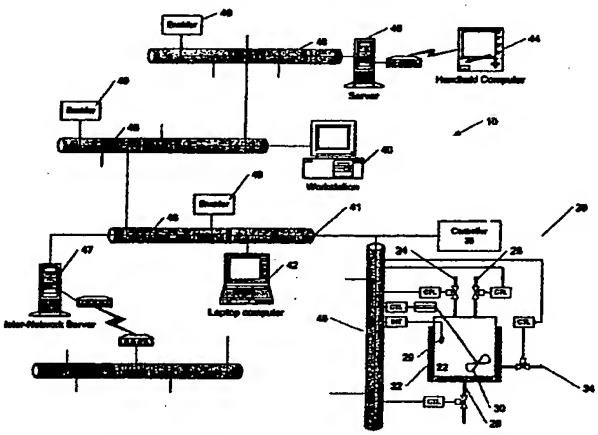
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHODS AND APPARATUS FOR CONTROL USING CONTROL DEVICES THAT PROVIDE A VIRTUAL MACHINE ENVIRONMENT AND THAT COMMUNICATE VIA AN IP NETWORK



(57) Abstract: The invention provides improved methods and apparatus for control using field and control devices that provide a virtual machine environment and that communicate via an IP network. By way of non-limiting example, such field device can be an "intelligent" transmitter or actuator that includes a low power processor, along with a random access memory, a read-only memory, FlashRAM, and a sensor interface. The processor can execute a real-time operating system, as well as a Java virtual machine (JVM). Java byte code executes in the JVM to configure the field device to perform typical process control functions, e.g., for proportional integral derivative (PID) control and signal conditioning. Control networks can include a plurality of such field and control devices interconnected by an IP network, such as an Ethernet.

WO 00/77592 A2

METHODS AND APPARATUS FOR CONTROL USING CONTROL DEVICES THAT PROVIDE A VIRTUAL MACHINE ENVIRONMENT AND THAT COMMUNICATE VIA AN IP NETWORK

Background of the Invention

This application claims the priority of the following United States Patent Applications: United States Patent Application Serial No. 60/139,071, entitled OMNIBUS AND WEB CONTROL, filed June 11, 1999; United States Patent Application Serial No. 60/144,693, entitled OMNIBUS AND WEB CONTROL, filed July 20, 1999; United States Patent Application Serial No. 60/149,276, entitled METHODS AND APPARATUS FOR PROCESS CONTROL ("AUTOARCHITECTURE"), filed August 17, 1999; United States Patent Application Serial No. 09/345,215, entitled PROCESS CONTROL SYSTEM AND METHOD WITH IMPROVED DISTRIBUTION, INSTALLATION, AND VALIDATION OF COMPONENTS, filed June 30, 1999.

The invention pertains to control systems and, more particularly, to methods and apparatus for networking, configuring and operating field devices, controllers, consoles and other control devices.

The terms "control" and "control systems" refer to the control of a device or system by monitoring one or more of its characteristics. This is used to insure that output, processing, quality and/or efficiency remain within desired parameters over the course of time. In many control systems, digital data processing or other automated apparatus monitor a device, process or system and automatically adjust its operational parameters. In other control systems, such apparatus monitor the device, process or system and display alarms or other indicia of its characteristics, leaving responsibility for adjustment to the operator.

Control is used in a number of fields. Process control, for example, is employed in the manufacturing sector for process, repetitive and discrete manufactures, though, it also has wide application in utility and other service industries. Environmental control finds application in

residential, commercial, institutional and industrial settings, where temperature and other environmental factors must be properly maintained. Control is also used in articles of manufacture, from toasters to aircraft, to monitor and control device operation.

Modern day control systems typically include a combination of field devices, controllers, workstations and other more powerful digital data processing apparatus, the functions of which may overlap or be combined. Field devices include temperature, flow and other sensors that measure characteristics of the subject device, process or system. They also include valves and other actuators that mechanically, electrically, magnetically, or otherwise effect the desired control.

Controllers generate settings for the control devices based on measurements from sensor type field devices. Controller operation is typically based on a "control algorithm" that maintains a controlled system at a desired level, or drives it to that level, by minimizing differences between the values measured by the sensors and, for example, a setpoint defined by the operator.

Workstations, control stations and the like are typically used to configure and monitor the process as a whole. They are often also used to execute higher-levels of process control, e.g., coordinating groups of controllers and responding to alarm conditions occurring within them.

In a food processing plant, for example, a workstation coordinates controllers that actuate conveyors, valves, and the like, to transport soup stock and other ingredients to a processing vessel. The workstation also configures and monitors the controllers that maintain the contents of that vessel at a simmer or low boil. The latter operate, for example, by comparing measurements of vapor pressure in the processing vessel with a desired setpoint. If the vessel pressure is too low, the control algorithm may call for incrementally opening the heating gas valves, thereby, driving the pressure and boiling activity upwards. As the pressure approaches the desired setpoint, the algorithm requires incrementally leveling the valves to maintain the roil of the boil.

The field devices, controllers, workstations and other components that make up a process control system typically communicate over heterogeneous media. Field devices connect with controllers, for example, over dedicated "fieldbuses" operating under proprietary or industry-specific protocols. Examples of these are FoxCom(TM), Profibus, ControlNet, ModBus, DeviceNet, among others. The controllers themselves may be connected to one another, as well as to workstations, via backplane or other proprietary high-speed dedicated buses, such as Nodebus(TM). Communications among workstations and plant or enterprise-level processors may be via Ethernet networks or other Internet Protocol (IP) networks.

Control device manufacturers, individually, and the control industry, as a whole, have pushed for some uniformity among otherwise competing communication standards. The Foundation Fieldbus, for example, is the result of an industry-wide effort to define a uniform protocol for communications among processor-equipped (or "intelligent") field devices. Efforts such as this have been limited to specific segments of the control hierarchy (e.g., bus communications among field devices) and are typically hampered by technological changes that all to soon render the standards obsolete.

Still less uniform are the command and operation of control devices. Though field devices may function at the direction of controllers and controllers, in turn, at the direction of workstations (or other plant-level processors), proprietary mechanisms within the individual components determine how they perform their respective functions. Even the commands for invoking those functions may be manufacturer- or product-specific. Thus, the commands necessary to drive actuators of one manufacturer will differ from those of another. How the corresponding commands are processed internally within the actuators differ still more (though, hopefully, the results achieved are the same). The specific programming codes used to effect a given control algorithm likewise differs among competing makes, as do those of the higher-level control processors.

Industry efforts toward harmonization of software for command and operation of control devices have focused on editing languages that define process control algorithms. This is distinct from the codes used to effect those algorithms within control devices and, rather, concerns software

"tools" available to users to specify the algorithms, e.g., editors including IEC-1131 standard languages such as Field Blocks, Sequential Function Charts (SFC), Ladder Logic and Structured Text.

Less concerted are industry moves to extend monitoring and limited configuration capabilities beyond in-plant consoles, e.g., to remote workstations. An example of this was the abortive Java for Distributed Control (JDC) effort, which proposed enabling in-plant workstations to serve web pages to remote Java bytecode-enabled client computers. The latter used the to web pages to monitor and set control parameters which the workstations, in turn, incorporated into their own control schemes.

An academic system along these same lines was suggested by the Mercury Project of the University of Southern California, proposing the use of a web browser to enable remote users to control a robotic arm via a server that controlled the arm. A related company-specific effort included that announced by Tribe Computer Works that allegedly enabled users to manage routers and remote access servers over IP networks using web browser software. See, "Tribe Defines Net Management Role For Web Browser Software," Network World, May 22, 1995, at p. 14.

Thus sets the stage for the present invention, an object of which is to provide improved methods and apparatus for networking, configuring and operating field devices, controllers, consoles and other control devices. A related object is to provide such methods and apparatus for process control.

Further objects of the invention are to provide such methods and apparatus as reduce the confusion, complexity and costs attendant to prior art control systems.

Related objects of the invention are to provide such methods and apparatus as can be implemented with commercial off the shelf hardware and software.

Still further objects of the invention are to provide such methods and apparatus as achieve confusion-, complexity- and cost-reduction without hampering manufacturer creativity and without removing incentives to development of product differentiators.

Summary of the Invention

The foregoing are among the objects attained by invention which provides, in one aspect, an improved field device for a process or other control system. The field device includes a virtual machine environment for executing Java byte code (or other such intermediate code) that, for example, configures the device to execute a control algorithm.

By way of non-limiting example, the field device can be an "intelligent" transmitter or actuator that includes a low power processor, along with a random access memory, a read-only memory, FlashRAM, and a sensor interface. The processor can execute a real-time operating system, as well as a Java virtual machine (JVM). Java byte code executes in the JVM to configure the field device to perform typical process control functions, e.g., for proportional integral derivative (PID) control and signal conditioning.

Further aspects of the invention provide a field device, such as a low-power intelligent actuator, that incorporates an embedded web server. This can be used to configure, monitor and/or maintain the device itself (as well as other elements of the control system) via a browser attached directly to the device or coupled to it over the network. To this end, the field device can incorporate a configuration editor, e.g., operating on a processor within the field device, that an end-user executes via the browser and web server.

Such a configuration editor can, in related aspects of the invention, be enabled or disabled depending on the environment in which it is used and more specifically, for example, the type of network in which it is incorporated. Thus, for example, the editor can be disabled when the field device is incorporated in a process control network that includes, e.g., an applications development environment suitable for configuration of the field device. Conversely, it can be enabled when the field device is incorporated in a network that lacks such a capability.

Still further aspects of the invention provide a field device as described above that includes an interface to an IP network, through which the device communicates with other elements of the control system. The IP network can be, for example, an Ethernet network. Moreover, it can be "powered," carrying electrical power as well as packets, datagrams, and other control or data signals. The field device, in related aspects of the invention, draws operational power, e.g., for its processor and other components, from such a network.

Yet further aspects of the invention provide a field device as described above that obtains configuration information and/or its network address from such an IP network upon start-up. To this end, on power-up or coupling to the network, the field device can supply an identifier (e.g., attained from a letterbug, assigned by a hub, or otherwise) to a DHCP or other server on the network. Once provided with an IP address, the field device can formally enter into the control network, e.g., by posting its characteristics to a network bulletin board, e.g., using a network enabler such as a Jini and/or JavaSpace server, or the like. Other network devices monitoring or notified via such a bulletin board can send configuration information to the field device or otherwise.

Still further aspects of the invention provide control devices, such as servers, control stations, operator consoles, personal computers, handheld computers, and the like, having attributes as described above. Such control devices can have other attributes, according to further aspects of the invention. Thus, by way of non-limiting example, they can provide web servers that collect process data from one or more control devices, generate source for operator displays, provide access to the control system, and host an applications development environment.

Still other aspects of the invention provide process, environmental, industrial and other control systems that comprise field and control devices as described above that are coupled via an IP network and, particularly, for example, by a powered IP network.

Additional aspects of the invention are directed to DHCP servers and network enablers (optionally, including web servers) for use in control systems as described above. Related

aspects provide such servers and enablers as are embodied in solid state technologies, e.g., with no moving parts.

These and other aspects of the invention are evident in the attached drawings, and in the description and claims that follow.

Brief Description of the Drawings

A more complete understanding of the invention may be attained by reference to the drawings, in which:

Figure 1 depicts a process control system 10 according to one practice of the invention;

Figures 2 and 3 depict more particular embodiments of a control system of the type shown in Figure 1;

Figure 4 depicts a native intelligent field device according to the invention;

Figure 5 depicts a native control device according to the invention; and

Figure 6 depicts a native intelligent positioner implementation according to the invention.

Detailed Description of the Illustrated Embodiment

Figure 1 depicts a process control system 10 according to the invention. The system includes networked control devices that monitor and control a hypothetical mixing process that utilizes mixing chamber 22, fluid inlets 24, 26, fluid outlet 28, paddle 30, cooler 32, and cooler inlet 34. Though illustrated and described below for use in connection with process control, those skilled in the art will appreciate that apparatus and methods according to the invention can be used in connection any industrial, manufacturing, service, environmental or other process, device or system amenable to monitoring or control (hereinafter, collectively, "control").

The networked control devices include actuators, such as the valves depicted as controlling inlets and outlets 24 - 28 and 34. A further actuator is shown controlling paddle 30. These and other actuators utilized by the control system are constructed and operated in the conventional manner, as modified in accord with the teachings hereof. The actuators operate under control of respective field device controllers, labeled CTL, that are also constructed and operated in the conventional manner to provide initialization, signal conditioning and communications functions.

Rather than using separate controllers CTL, the actuators can be of the intelligent variety and can include integral microprocessors or other digital data processing apparatus for control, initialization, signal conditioning, communications and other control-related functions. For sake of convenience, the label CTL is used regardless of whether the control-related functionality is integral to the actuators (e.g., as in the case of intelligent actuators) or otherwise.

Illustrated sensor 29 monitors a temperature, level or other characteristic of fluid in chamber 22. The sensor 29, as well as other sensing apparatus utilized by the system, are constructed and operated in the conventional manner known in the art, as modified in accord with the teachings hereof. They can be coupled to the control network via a transmitter or other interface device INT that, too, is constructed and operated in the conventional manner, as modified by the teachings hereof. The interface devices facilitate initialization, signal conditioning and communications between the sensors and the control system. As above, one or more sensors can be of the intelligent variety, incorporating integral microprocessors or other digital data processing capabilities for initialization, signal conditioning, communications and other control-related functions. Here, too, the label INT is used in reference to the control-related functionality, regardless of whether embodied in an intelligent transmitter or otherwise.

The networked control devices include one or more controllers 36 that monitor and control respective aspects of the hypothetical mixing process in the conventional manner, as modified in accord with the teachings hereof. The controllers can comprise mainframe computers, workstations, personal computers, special-purpose hardware or other digital data processing

apparatus capable of performing conventional monitoring and control functions. Preferred controllers are constructed and operated in the manner of the CP control processors commercially available from the assignee hereof, as modified in accord with the teachings herein.

The control system 10 includes a variety of devices that serve as user interfaces and that provide configuration and/or control functions, all in the conventional manner as modified in accord with the teachings hereof. Illustrated for these purposes are workstation 40, laptop computer 42 and handheld computer 44. These devices can provide configuration and control functions directly, as in the case of workstation 40, or in cooperation with server devices, e.g., as in the case of handheld computer 44 and server 46. Apparatus 40 - 44 can couple with the control network directly, e.g., via bus or network connection, or indirectly, e.g., via satellite, wireless connection or modem connection.

The control devices 36 - 46, CTL and INT, collectively, referred to as "native" devices, are coupled for communications via a medium that permits at least selected ones of the devices to communicate with one another. To this end, in the illustrated embodiment those devices are coupled via one or more networks 48 that are, preferably, IP-based such as, by way non-limiting example, Ethernets. The network(s) can include, as indicated by the multiple segments shown in the drawing, multiple segments such as various wide and local area networks. They may also include high and/or low bandwidth components, such as phone lines, and low and/or high latency components, such as geosynchronous satellites networks.

Figure 2 depicts a more particular embodiment of a control system 50 of the type shown in Figure 1. The system includes an enterprise server 52, a first thin client 54, plant server 56, a second thin client 58, a controller 60, a Java-enabled field device 62, and one or more field devices 64, coupled to one another, e.g., in the manner illustrated, by one or more networks 66, 68.

Native enterprise server 52 (corresponding, by way of non-limiting example, to server 46) comprises a mainframe computer or engineering workstation that executes enterprise-level

applications such as, by non-limiting example, those for financial, asset planning and procurement, distribution and/or human resources. In addition, it supports web serving, as well as optional object, relational and other database management systems. Server 52 executes Windows NT, Solaris or another conventional commercial or proprietary operating system. It is also equipped with a Java Virtual Machine (JVM), e.g., capable of executing Java virtual machine instructions (bytecodes), of performing remote method invocation (RMI), and of supporting Jini networking. The enterprise server 12 can be coupled to further networks, e.g., to the Internet, as shown, in any manner known in the art.

Native thin client 54 (corresponding, for example, to handheld computer 44) provides similar functionality as server 52, though its actual processing activity is limited to user input and output. Application processing, such as financial, asset planning and procurement, distribution and/or human resources, are performed on behalf of client 54 by a server, such as server 52. The operating system and JVM functions may be embedded in the conventional manner of a thin client. The thin client 54 is coupled to the enterprise server 52 over a business information network 66 (corresponding, for example, to network 48), typically, an Ethernet or other IP network, configured in a LAN, WAN or the like.

Native plant server 56 (corresponding, by way of non-limiting example to workstation 40) comprises a plant control console or engineering workstation modified in accord with the teachings hereof, executing plant-level control applications including system, process, engineering, plant information, configuration, lab quality, maintenance, resource and documentation applications of the type known in the art . Like enterprise server 52, plant server 56 can execute Windows NT, Solaris or other conventional operating systems. It is also preferably equipped to execute a Java Virtual Machine as described above. Plant server 56 is coupled to the enterprise server 52 and thin client 54 over the business information network 66.

Native thin client 58 provides similar functionality as server 56 though, again, relies on that (or another) server to perform most processing activity. As above, the operating system and JVM functions may be embedded in the conventional manner of a thin client. The thin client 58 is

coupled to the plant server 56 over a control network 68 (corresponding, for example, to network 48), e.g., an Ethernet.

Native controller 60 (corresponding, for example, to controller 36) executes control algorithms to control associated non-native field devices 64, e.g., via any variety of commercial and/or proprietary field bus 70 hardware and protocols. Where processing resources are limited, the controller 60 utilizes a embedded operating system that supports web serving and the JVM. The controller 60 is coupled to the plant sever 66 and to the thin client 58 via control network 68.

Native field device 62 is a sensor, actuator or other field device. The illustrated device is of the intelligent variety, including processor (not shown) and operating system. It can be of the type commercially available in the marketplace, as modified in accord with the teachings hereof. The illustrated device supports web serving and JVM, as described above. The device 62 provides information collection and control functions, as illustrated.

Figure 3 depicts another a more particular embodiment of a control system 50 of the type shown in Figure 1.

Referring to Figure 1, the illustrated control system 10, 50 uses web browsers, Java, Jini, JavaSpaces, TCP/IP and other technologies normally associated with the Internet and the world wide web to create a self-defining control network that minimizes complexity while emphasizing the portability and re-usability of the user's application. These technologies are advantageously employed to eliminate proprietary hardware and software to preserve the user's investments; eliminate network configuration and system management through self-configuring networks; increase the user's choice of algorithm suppliers by implementing control in Java; preserve the user's applications through hardware and system software changes through the use of Java and Web Browsers for process displays; minimize the wiring costs; reduce maintenance by making intelligent field devices a practical reality.

In addition to web technologies, the illustrated control system 10 uses an object location service, a commercial messaging service, and standard networking equipment, such as hubs, routers,

network interface cards, where applicable. It also relies on common standards, where applicable, such as 802.3, Internet RFCs, the IEEE 1451 sensor standards. The system 10 supports a heterogeneous computing environment and utilizes industry standards (e.g., Microsoft standards) to provide communications between the native control components to the business and desktop environments.

1 Device Hardware

1.1 Platform-Defining Control Devices

In the illustrated embodiment, native control devices such as controllers 36, 60, workstation 40, servers 46, 52, 56 providing a platform for the control system typically include a central processing unit (CPU), memory (RAM), network support hardware, access to permanent storage, an operating system, and a Java Virtual Machine (JVM) including the TCP/IP suite. In addition, the devices include web server software, e.g., software of the type that serves graphical "web" pages in response to requests by other devices. Configurator software can be provided, as well, permitting each device to configure the control system or selected portions of it. Those skilled in the art will appreciate that not all of these components need be included in all native control devices, e.g., some commercially available JVMs can serve as an OS themselves.

Process control object (PCO) software provided on the platform-defining control devices comprise a collection of data and methods implemented in Java and executed on the native control devices' JVMs to perform typical process control functions, by way of non-limiting example, signal conditioning and PID control. Likewise, station management object (SMO) software on the devices comprise data and methods that allow the device to report its health, performance, and other status information in a uniform manner. The SMO software can implement SNMP or the equivalent Java functionality.

Software is also provided on the platform-defining controls devices for messaging services for data transfer and alarm/event notification, as well as software comprising system management pages. Each control device may include additional software, of course, depending on its functionality.

1.2 Field Devices

Native intelligent field devices typically include a low power CPU, e.g., a NetARM or Java Chip; a real-time OS like VxWorks, QNX, PSOS; RAM; FlashRAM to serve as permanent storage; ROM/EEPROM to serve as the home for the OS and other permanent software; an Ethernet interface; power from the Ethernet interface (in the event a powered Ethernet network or hub is used) or otherwise; a sensor interface , e.g., IEEE 1451.1 and 1451.2; JVM; a web server, and a device specific configurator servlet. A field device so constructed can be configured and monitored via a lightweight web browser, e.g., handheld computer 44, coupled to the device over the network 48.

The field devices can include serial interfaces to allow the attachment of these devices to HART, FoxComm, Profibus and other networks operating under a protocol different from that of network 48. Combined with appropriate software, these devices provide the user with a single transmitter suitable for use on any field network.

One configuration of a native intelligent field device including the elements described above these elements is shown in Figure 5. Those skilled in the art will appreciate that other configurations can be realized, as well, in accord with the teachings hereof.

1.2.1 Stoic Sensors

The control system 10 can include one or more stoic sensors, not illustrated, that constitute minimal sensing elements. Typically, these are simply silicon chips that are packaged to allow them to sense the process environment and that have only enough electronics to relay measurements to a pair of wires that carry the raw signal to another device for processing. Some stoic sensors generate a signal without external power; others require some excitation. In a preferred embodiment, native devices 36 - 46, CTL and INT according to the invention support the attachment of many stoic sensors.

1.2.2 I/O

1.2.2.1 Overview

Some I/O devices, e.g., thermocouples, RTDs and analog transmitters, may not use a transmitter compatible with the network 48 but, rather, send raw sensor output to a multiplexor for processing. To accommodate these devices, the system 10 includes two types of intelligent I/O cards: network I/O cards supporting IP and an API, and native I/O cards that support the protocol of network 48 (i.e., the "native" protocol) directly.

Network I/O cards are coupled directly to network 48. Native control devices are IP enabled and support the proprietary API of the cards, thereby, permitting reading and writing of their I/O registers. The cards' respective APIs support retrieval of data in several formats (raw counts, linearized counts, engineering units, etc.), as well as the assignment of simple configuration information to those registers, if the device supports such functions. To this end, the native devices utilize I/O classes that corresponding to the native I/O protocols. Redundant I/O devices can be physically interfaced transparently through a single I/O card or through multiple independent network I/O cards. Logically, PCOs support at least the use of multiple independent cards. Alternative I/O features may be used in addition. Native I/O cards are substantially similar to native-enabled transmitters, except for packaging and scale. These I/O cards may be considered as small multi-loop controllers or multiplexors.

1.2.2.2 Foreign Device Integration

Networked I/O cards are utilized with devices that cannot otherwise directly interface with the network 48. The cards, in this case, provide an interface that permits at least reading and writing of relevant device values, which can be stored internally to the card and accessed via its API. Foreign devices that are control systems in its own right typically maintain objects or other data structures representing process values and associated data. A preferred interface to these "devices" are through objects that wrap the foreign function blocks with the services expected of native devices, , e.g., detail displays and configurators, which are accessed in exactly the same manner as native ones of those same services. For example, a name service is be able to locate foreign blocks and the native API of system 10 permits querying those blocks for values.

1.3 Native Controllers

In a preferred embodiment, all native devices may be used as controllers, though, dedicated I/O-less native controllers, such as controllers 36 and 60, are typically required for unit-wide operations. Thus, for example, control can also be provided by personal computers, workstations and servers of the type shown as elements 40, 42, 46, in Figure 1. Native controllers, e.g., 36, 60 are of the same general design as native boards used in the transmitters and actuators. However, they support a faster CPU clock cycle and more memory (volatile and non-volatile) than their smaller siblings. High performance native controllers have a still more powerful CPU, more RAM, and maybe a removable FlashRAM card for bulk storage and backup. Like the native transmitters and actuators, these native controllers can receive power from a powered Ethernet connection or otherwise. Native controllers can be used in redundant and non-redundant configurations. Since the I/O is independent of the native controller, redundancy can be implemented in a number of manners, e.g., hardware fault-tolerance or transaction based synchronization between stations with clustering.

1.4 Native Compliant Workstations

Native workstations, e.g., such as element 40, can be constitute web browser-enabled devices that communicate with web servers, such as element 46, that actually collect the process data from other native devices. In addition, the workstations can consist of a flat panel display, OS and web browser in ROM (or on a memory card), web page (process graphic) database/cache, connections for optional annunciator keyboards, option for wireless Ethernet, and, optional, battery operation. Data supporting the operator interface comes from native devices directly.

1.5 Native Web Server

If a generic web browser-enabled device is the operator's console, a native web server, such as servers 46, 52, 56 sources the operator displays. This can be implemented with redundancy using technologies NT's clustering capabilities, or the like. Additionally, a native web server provides access to a illustrated control system 10 to users not physically attached to the illustrated control system 10, e.g., as illustrated with respect to elements 44 and 46 of Figure 1. In this mode, it centralizes data requests to maximize efficient use of communication resources.

1.6 Native Enablers

1.6.1 The illustrated control system 10 includes native enablers 49, such as Jini and JavaSpaces devices and Solid-State DHCP servers. These enablers, which are optionally redundant, are preferably fully solid state with persistent memories. They do not use batteries for their persistent memory, though they may well plug into a wall outlet for power in non-industrial environments. The Jini and JavaSpaces serve as community "bulletin boards." These are used to notify system monitors that native devices have been added to or removed from the system. The system monitors respond to such notices by triggering appropriate actions.

The illustrated control system 10 assumes that each native device is able to acquire an IP address when it comes on-line. To minimize configuration, a DHCP server, illustrated for example by element 49, is required to furnish these addresses. To ensure maximum availability of this critical server, it is preferably a solid-state device and, optionally, includes a redundant partner DHCP Server. A particular native DHCP server provides addresses for a portion of the network 48. It obtains its configuration by notifying the system 10 that it has come on line.

Those skilled in the art will, of course, appreciate that IP addresses can be selected by mechanisms other than DHCP servers.

1.6.2 Native Internetwork Server

An internetwork server 47 is used to link separate control systems networks 48. It provides a platform for inter-network communications, controlled access to data, name conflict resolution, and other services need to efficiently, securely, and transparently accomplish the connection of one or more illustrated control systems to another.

2 SubSystems

2.1 Software Downloads

The illustrated system permits the electronic downloading of software and/or licenses for execution on the native devices. This includes native software objects, updates and/or licenses for same. To ensure proper operation of the process facility even in the event of insufficient licenses, downloaded software registers with the system monitor and declares whether or not it is

licensed. In the case of insufficient licensing, the system monitor's notifies the customer appropriately.

In instances where appropriate native software modules and authorizations are in place, the system 10 can access a download site, e.g., an Internet e-commerce site, and determines if updates are available. If so, it notifies the user and invites him/her to request downloading of the upgrade. Such an e-commerce or other web site can also provide configuration tools that allow the user to design, implement, and test a new PCO block. This allows the user to access complex and/or rapidly improving software tools without having to maintain them locally. For this purpose, the use is charged, e.g., a modest access fee.

2.2 Security

The system 10 includes a native security system to control access to managed objects based on the type of access, e.g., (i) extra-network, i.e., users who are accessing the illustrated control system 10 from a non-native station; (ii) inter-network, i.e., users who are operating from a native workstation on a different physical native network; and (iii) intra-network, i.e., users who are operating from a native workstation within the particular native network.

Secured extra-system access is provided through a native secure web server, e.g., server 46, that permits dial-up, network, and other remote access and that supplies and defines the permitted extra-network access, including the API available to applications hosted outside the network or on the server. See, for example, elements 44 and 46 of Figure 1. Access is controlled by user name, by user location (IP address or workstation name depending on the network), and by the type of the targeted object.

Inter-system access is provided by a gateway device, such as server 47, that permits the secure transfer of data. This device negotiates secure access, deals with name conflicts between systems, and provides support for various physical media A pair of such devices are provided to account for situations where the source is local to the sink. In a preferred system, the server 47 or other gateway encrypts the data so that others cannot read it. Likewise, it authenticates message sources to verify that they are coming from a matching device. A preferred gateway

minimizes the number of packet transfers so as to minimize delays over slow or high latency links.

Secured intra-system access is controlled based on the user and the workstation, leveraging the Java security model and other security models to the extent possible. The native security system authenticates users, e.g., regardless of whether they are attempting access from operator's console or via an application program.

The native security system manages access to objects, including PCO attributes (i.e., variables and/or parameters of PCOs), system monitors, and native messages. It allows applications to add objects to the list of managed objects. Read and write access is preferably controlled separately, i.e., as if a single PCO attribute was two separate objects.

The security model is based on a lock and key approach. Each PCO attribute, for example, is assigned to one of a large number of locks. The user is given different keys for read and write access to object attributes. This key may be modified according to the type of native workstation used for access. The key is passed to the object attribute when the object attribute is accessed. If the access is a connection, it need be passed only once. The object attribute compares the key to its lock and allow or deny access as appropriate.

A software tool is supplied with the applications development environment (ADE) or configurator that allows identification of users; access points allowed by the user; object attribute groups accessible to the user; and access type (read or write) to the object types. The object attribute groups define collections of object attributes via similar security access guidelines, e.g., alarm limits might be in one group and tuning parameters might be in another. The security model is embedded in the APIs providing access to the secured objects. This assures that that access is granted only after the native security system has performed the necessary verifications.

2.3 Maintenance and Support

The illustrated control system 10 supports the on-line upgrade of all application software from remote and local workstations. Application software, for purposes of this discussion, includes the

system monitor, PCOs, and other Java-based code. Native diagnostic and maintenance tools work over user-supplied or other IP-based networks, providing that they allow services such as the world wide web to operate over them to external sites. For networks that do not permit this, the illustrated control system 10 provides modules that support direct connections to a native support center via dialup analog phone lines, and dialup high speed lines, e.g., ISDN and ADSL. Native maintenance, including software updates, operate over these connections; hence, it is not necessary for a person to be physically present to update system or application software.

The illustrated control system IO includes diagnostics to track problems from the hardware upwards. Maintenance staff can run in parallel diagnostic versions of the software. Such software running in open loop or switched into operation as necessary is believed particularly advantageous for support purposes.

2.4 Configuration

An application development environment (ADE) coordinates configuration in the illustrated control system 10, e.g., except for devices (such as certain transmitters and positioners) where use of an ADE is not advantageous and for which internal configuration is preferred. Characteristics of the ADE are its use in configuring/building control applications including control schemes, displays, historians, etc.; support of multiple simultaneous users; support of remote, concurrent development; support of change tracking and change management including task based activity traces, summary reports, change annotation, approval cycles, and user identification; a human interface component that runs in any web browser coupled to the system; a server component that runs on any user-supplied system; permanent storage comprising a commercial database for which a JDBC implementation is available; allowing the definition of configuration object templates; allowing the instantiation of configuration object templates into configuration objects; allowing the user to add and remove editors for configuration object components in real-time; limiting the user's access to configuration capabilities in the various pieces of equipment; and supporting application distribution along with verification of download permissions.

With respect to the support of multiple simultaneous users, system editors provide object-based "check-out" and "check-in." No other user is allowed to edit a checked-out object or any of the objects that it contains. As an alternative, editors support the concept of task-based configuration activities and the related "check-out" and "check-in" facilities required to make it work. When an object is checked out, it is assigned to a task. Objects are not checked back in, tasks are. A build manager has the responsibility of integrating the tasks.

The native ADE delivers lower engineering and maintenance costs through the unification of application development, preservation of application expertise, reduction of application development effort, and the deployment of developed applications. Is it based on industry standards (e.g., IEC 1131 and IEC 1499) to preserve the user's investment in training and applications. It also produces appropriately formatted Java class for execution in native devices. Since the native ADE produces output that can be read and loaded into a JVM and since the native control devices include JVM, one control configurator can configure all native devices.

In a preferred embodiment, the ADE imports configurations from legacy systems; supports the use of third-party control algorithms; and supports both bulk building of control configurations and on-line changes with validation.

Configuration is not limited to implementation on a web browser, though this can be advantageous since it allows configuration from many types of device without installation of special software, i.e., it provides a thin-client configuration mechanism. For device configuration, there are two cases: a device used in illustrated control system 10 and a device used outside illustrated control system 10. For devices used outside illustrated control system 10, the configurator is placed in the device so that it can be configured even without the native ADE. This on-board configurator allows the device to be configured for use with Profibus, Foundation Fieldbus, on 4-20ma wires, and other industry-standard or proprietary networks. For devices used in the native environment, (i) a copy of the configurator is available from a native web server for off-line configuration and download, and (ii) the on-board configurator is disabled to prevent changes in the configuration except through the ADE.

2.5 Human Interface

The illustrated system's primary human interface (HI) device is a web browser. The current range of devices executing these spans cellular phones to mainframe computers. The native HI is multilingual, i.e., it supports the presentation of information in a default language and one or more secondary languages simultaneously. All standard native applications support text substitution based on the currently selected language. Error messages are likewise in the currently selected language.

2.6 Process Control

Process Control is implemented using process control objects (PCOs) running in an execution environment consisting of a JVM and any associated applications required to load and execute the specified control strategies. Control strategies are specified in terms of PCO composites. PCOs are Java classes designed to be modular and easily upgraded even during operation. The native PCO configurator, from a high level view, creates instances of these classes, connects them as necessary, and installs them in particular devices.

Process control objects consist of two user-available types: blocks and composites. PCO blocks are, from the user's point of view, similar to conventional function blocks. Likewise, composites are similar to conventional collections of function blocks. Distinguishing external features of PCOs include the fact that changes involve the addition/deletion of PCOs and cause new versions to be loaded into the targeted native device. The new version runs in parallel with the old version until the engineer decides that it is operating properly and "swaps" the control from old to new.

Composite PCOs may span stations transparently. PCOs are bound to stations very late in the configuration process and may be migrated from one station to another at any time. Further, PCOs from different sources may operate in the same device if the configurator supports the use of multiple PCO libraries.

A native PCO configurator supports multiple libraries of PCOs from multiple vendors; permits the creation of composite PCOs from other PCOs; permits the use of composite PCOs as

templates, with a composite PCO definition specifying which fields is altered in a template; permits the assignment of PCOs to physical devices; provides IEC 1131/1499 influenced view of the configuration process, i.e., support for Logic Diagrams, Structured Text, Sequential Function Charts, Function Blocks, and PCOs, while producing Java byte code as its output.

The creation of composites includes the raising of internal (deeply buried) names to the top level of the composite PCO. For example, a cascaded Temperature/Flow loop might have the temperature and flow measurements referenced as TC_CASCADE:PRIMARYFLOW and TC_CASCADE:SECONDARYFLOW or as the longer fully qualified name. The configurator works both off-line in a bulk creation mode and on-line in an individual correction mode.

2.7 Communications: Object Location, Message Transfer, and Data Transfer

The concepts of object location and data/message transfer are closely bound. A process control system imposes significant quality of service requirements on any services used to locate objects of interest and to acquire values from or make changes to those objects. The illustrated system includes an object location and data transfer service that provides a communication model definition, security, object location services, network types requiring support, quality of service, APIs (Java and non-Java), maintenance and upgrade strategy, and interoperability with existing control systems. In addition, the communications system addresses the particular needs of process data transfer.

2.8 Critical Applications

2.8.1 Time Synchronization

The illustrated control system 10 supports time synchronization to the millisecond in each station on the network 46. Where equipment configuration renders this impossible, time synchronization to 50 ms is provided.

2.8.2 Alarm and Message Management

The illustrated control system 10 provides a facility that centralizes viewing, recording, organizing, and categorizing the messages generated by the system monitor, the operator action journals, the PCOs, and other systems that record textual information. This application is the

basis for alarm management strategies including inferential alarming and alarm prediction. However, the message management facility is not in and of itself an alarm historian. Rather, it relies on a native historian to provide the long-term storage of this information.

2.8.3 Historian

The native historian provides the fundamental data collection, reduction, and archival services. The data collected includes process values and messages from various applications within the illustrated control system 10. In addition, it provides historical data in support of several other common applications – typically known as plant information management system (PIMS) programs and trend window support. The historian is capable of exporting its data to user-selected databases.

2.8.4 Plant Information Management System

These applications include a simple and easy to report writing facility, a calculation facility that can use historical and real-time data to compute new values, and a desktop visualization system that uses the historians data instead of real-time data. The desktop visualization system uses the native graphics capabilities to connect the graphics to the historian and to allow the operator to "move" the entire graphic backwards and forwards through the historical data. The values calculated by the calculation facility are stored in PCOs that represent the data and generate appropriate alarms. The report writing facility supports shift reports with calendar adjustments as a simple to use feature. In addition, it facilitates the definition of ad-hoc reports using a page layout metaphor.

3 Interoperablity

Interoperability of the illustrated control system IO with legacy process control systems can be facilitated by adding a networked I/O module to the I/O chain of existing I/O cards, adding a native device integrator to talk to other networks of devices, or utilizing a native API for Microsoft- and Solaris-based applications.

4 Operating the Control System

4.1 System Startup and Management

4.1.1 Device Identification

Each native device is assigned a name that is used to identify the device and obtain any configuration information that it needs. Embodiments of the invention use alternate approaches to identifying a device, including using a hub that is configured by the user to assign a name to each of its ports, e.g., using letterbug or software configuration; installing a letterbug, e.g., on the workshop bench, which tells the device its name; using a PC or handheld computer, e.g., on the workshop bench, to give the device its name; or using soft letterbugs.

4.1.2 Address Acquisition

After a native device has its name, it acquires its IP address from its environment. The illustrated embodiment uses DHCP for this purpose. Preferably, the DHCP server is a native device, such as enabler 49, though other DHCP servers can be used, as well.

4.1.3 Registration of the Device

Once the device knows its address, it registers its characteristics on a native bulletin board, which can be colocated with the DHCP server, e.g., in an enabler 49. Typically, many software services in the system 10 register with the bulletin board for notification of additions of native devices. Upon registration of a device, these services are notified and enter into relationships, if any, with the new device. A typical situation would involve notifying a system monitor process that a native device is active. That process then updates its web page with information about this device. In a preferred embodiment that utilizes a hierarchical network, each bulletin board is covers a given "territory" or region of the network.

4.1.4 Configuration Acquisition

If a device is unconfigured, it marks the "Configured" attribute on its bulletin board entry as "False". Any system monitor receiving device notifications to this effect generates an alert. Once a device has its name, it is able to communicate with any networked supply of non-volatile storage. This is particularly useful in the case of devices with limited in device non-volatile storage, whom a system configuration utility notifies of the location of configuration information. Devices that can retain all of their configuration information can start up using that

configuration. Actions taken during start-up, e.g., taking PID algorithms out of hold, are configurable.

4.1.5 System Monitors

A system monitor monitors the health of system components (hardware and software), monitors the health and operation of user applications, provides configuration information to devices that request it, and monitors the licensing of software and upgrades for purposes of alerting the appropriate groups. The system monitor supports standard SNMP agents and any native specific system management protocols.

The predominant source of information used by the system monitors are the bulletin boards which, as noted above, are used by the native devices to record their current state of operation. As devices are added and removed from the bulletin boards, the system monitor displays that information. The information posted on the bulletin boards includes diagnostic information. A system monitor not only displays this information but also allows the user to access and operate any embedded diagnostic displays and operations.

4.2 Device Configuration and Management

4.2.1 Device Startup

Initial start-up of a native device involves the following steps:

1) Install the device on the network 48 and turn on the power as required by the device.
2) The device obtains an IP address from a DHCP server 49 on the network 48.
3) The device registers with the native name service.
4) The device waits for a configuration.
5) The device is now ready for configuration.

Restart of a native device is effected by the steps of

1) Install the device on the network 48 and turn on the power.
2) The device obtains an IP address from a DHCP server on the network.
3) The device registers with the bulletin board.

4) The device begins operation according the configuration stored in its non-volatile RAM. This RAM may be a network resource that it must recover. If that resource is unavailable, the device is essentially unconfigured and the above procedure is followed.
5) The device is now ready for configuration and operation.

4.2.2 On-Line Configuration

The steps involved in on-line native device configuration are:

1) The engineer starts a web browser on workstation 40, handheld computer 44 (wireless or otherwise), a PC, a native workstation, etc.
2) The default home page on the browser is cached and contains an applet that locates the native bulletin board and raises the initial web page that lists the services available from that workstation.
3) The engineer selects the ADE and a native device of interest.

If the device is on a native network 48, the native configurator (ADE) is used to make the changes. In this case, the web browser is pointed at the web server that hosts the ADE. This approach eliminates conflicts between field change and database changes. The configuration of non-native devices is provided by a service that encapsulates the native commands and passes them through a native device to the actual foreign device.

The steps in PCO configuration are

1) As in the off-line mode, the engineer accesses the ADE and makes necessary changes.
2) Once the changes are complete, the engineer tells the new objects to begin execution in "open loop" mode, i.e., their outputs are not allowed to go to the field or to any other display station other than the one used to configure them. (For examples, their names are not registered with the native name service.)
3) In the open loop mode, the engineer can "Verify and Validate" his new control scheme by viewing the detail displays for the new and the old objects "in parallel". The set of available detail displays may include special displays for evaluation of parallel composites.

4) Once the configuration is complete, the device records its new configuration in its local non-volatile RAM. Optionally, it updates a remote configuration database or just set a "changed configuration" status indicator much like conventional intelligent transmitters.

5 Native Software Subsystems

5.1 Security

5.1.1 Object Access

5.1.1.1 Locks and Keys

As noted above, the proposed security mechanism is a lock and key. In this model, the attributes of an object that are available to be manipulated are assigned a lock number. An application registers with the security system at start-up to obtain a key. The value of the key depends on the effective user id (name) and the station on which the application is running. The key is never visible to the user, so he/she cannot alter it. This key is available to any library that uses the security system to ensure proper access.

5.1.1.2 Security Database

A security database contains a list of users. For each user it allows the specification of a master key, i.e., a bit array with a bit set for each lock the user is allowed to unlock. There is one master key for each object type supported by the security system. By default, the security system supports the creation of master keys for the system monitor, process data (PCO attributes), and native messages (events). The database also allows modifications to the master key for a user based on the station used to perform an operation. These exceptions can be expressed as IP addresses, station names, or by the type of access being used (e.g., extra-system, inter-system, or intra-system).

5.1.1.3 Operation

The security system supports caching the security system database on specific servers to improve redundant operation and to speed key queries. This is generally not a significant issue since the query is done once per application. Each time an operation is attempted or, in some circumstances, once per connection, the key is passed to the target object along with any other information required by the object to achieve the user's request. It is the responsibility of the object to match a bit in the key with the lock on the attribute or method being altered. If there is a

match, the operation proceeds. If the match fails, the operation is aborted and the calling application is notified of the failure. If the lock is zero, the operation is allowed to continue.

5.1.1.4 Use

According to a preferred practice, the illustrated control system 10 supports at least 256 individual locks. This allows the application engineer to define 255 groups of objects for each type of object being accessed.

5.1.1.5 Impact on PCOs and Other System Components

The value record of a PCO attribute includes two keys: one for read access and one for write access. A default value for the key can be inherited according to rules concerning the use of the attribute, e.g., all alarm attributes have an assigned group by default. A system monitor interface implements the same mechanism.

5.2 Maintenance and Support

5.3 Configuration

A configuration object (CO) is a collection of objects plus the editor used to manipulate them. A Configuration Object Template (COT) is a CO that has not been instantiated. The application development environment (ADE) is the environment for configuration object editors. It shows the user the existing COs in the CO being edited and a list of available templates. It provides the mechanism for adding objects to a CO by instantiating a COT and for removing COs. A Configuration Object Component (COC) is any of the objects found in a Configuration Object. Typical components are: PCOs, Process Graphics, Reports, Historian Configuration Objects, etc. The ADE has two components: the human interface and the server. Several human interfaces can access the same server at any time.

When a Native device is placed on-line, its internal configuration services are disabled. All configuration instructions come from the application development environment. Where on-line configuration is permitted, the system monitor or other system component flags any configuration mismatches.

Invoking the ADE is tantamount to starting the editor of a particular top-level CO. The ADE presents all of the available COs within the selected CO and all of the available COTs. The ADE provides a mechanism for manipulating those objects and for adding new objects by instantiating COTs. Since the objects within the ADE are also Configuration Objects, i.e., they are themselves collections of objects with associated editors, configuration involves selecting an object, invoking its editor, and making changes. Each editor knows how to store its object's data in the persistent storage.

The ADE preferably includes a system editor, a control editor, a graphic editor, a message manager editor, a historian editor. All editors support the use and creation of templates by the user. The also support the assignment of their objects to particular stations in the native network. Not all objects require assignment, but the editors are responsible for it.

5.3.1 System Editor

The system editor allows the user to configure a logical arrangement of native device objects. The only devices it configures are those that are native-enabled. If a device, such as Networked I/O, does not support Native, this editor is not interested in it. The following table lists those devices, what they do, and what needs to be configured.

Device Type	Description	System Editor Configured Attributes
DHCP/Bullet in Board Server	These devices allocate IP addresses to stations and act as the bulletin board for the stations that ask for addresses. They are not configured to know what devices might be attached.	Name and association with a Web Server
Native Controllers	These devices act as PCO platforms.	Name and association with a DHCP/bulletin board Server (which may be a Web Server)
Native Workstations	The devices that supply the human interface to the process.	Name and association with a DHCP/bulletin board Server (which

		may be a Web Server)
Native Web Servers	This devices host Native Applications and graphics. Native Workstations may also be Web Servers.	Name

As with the other editors, the system editor is object oriented and template driven. The user is allowed to select an object type from a palette and attach it to the appropriate stations. Individual Device Editors are provided for each of the device types.

5.3.1.1 DHCP/Bulletin Board Editor

If the device is assigned to another DHCP/bulletin board device or a web server, the editor allows the user to specify the number of IP addresses that this device needs to be able to supply below it. In this case, when the device starts up, it gets its own IP address and IP address range from the Web Server or DHCP/bulletin board device to which it is assigned. If it is not assigned to such a device, the editor allows a range of IP addresses to be assigned to it. It is assumed that the device gets its own IP address from another, non-Native DHCP Server.

5.3.1.2 Native PCO platforms

Each device type on the network is provided a specific editor to control its specific configuration. Typical devices are: transmitters, actuators, controllers, etc. These editors are available for off-line configuration. In this case, the system monitor downloads any approved configurations at the next start-up. The editors may also be available on-board the device (e.g., for fixed function devices such as single station controllers), which allows direct configuration through a web browser. The on-board facility is automatically disabled once the device is placed on a native network. This ensures database consistency while allowing the use of the device on non-native networks. In the illustrated embodiment, device specific configurators are provide in ADE add-in and on-board, preferably, with the same code would be available and used for both.

5.3.1.3 Native Workstations

The configuration of this device includes user name(s) allowed: read only operation; full fledged operator; and so forth.

5.3.1.4 Native Server

The configuration of this station is a range of addresses if it is acting as a DHCP server.

5.3.2 The Control Editor

The Control Editor implements PCOs plus four of the IEC 1131-3 languages (LD, ST, SFC, and FB). Any control schemes built in those languages are translated into a Java file, compiled, and downloaded. The IEC 1131-3 concept of configurations, resources, programs, and tasks are translated into the native environment that consists of set of native platforms. In addition, the Control Editor supports defining of control schemes using PCOs in the same manner that it supports the use of IEC Function Blocks; supports the assignment of control entities to specific PCO platforms; the importation of instruments lists to generate, and PCOs.

5.3.3 The Graphic Editor

The Graphic Editor support all of the human interface (HI) functionality and graphics typical of this type of editor.

5.3.4 Application Configuration

Applications can only be assigned to native web servers.

5.3.4.1 Historian Object Editor

A historian object when found in a CO is a representation of a portion of the full historian configuration, e.g., a slice of the historian. The historian object consists of the name of the historian to receive the data and a list of PCO attributes. Each attribute is associated with certain critical information related to the historian, e.g., change deltas, sample rates, reduction operations, archive characteristics, etc. Starting the historian editor resulting in showing the full configuration of the historian(s) to which this object is assigned.

5.3.4.1.1 Historian Data Definition Object Editor

One historian object exists for each historian in the system. These objects are created automatically if a historian data definition object is created and assigned to a historian. The data in this object controls the general function of the historian, e.g., where it runs in the system, what its archive policy is, what its data reduction policy is, etc.

5

5.4 Process Control

5.4.1 Definition of the Process Control Domain

The process control domain is usually divided into three primary control styles: (i) analog control (continuous), (ii) logic control (ladder logic), and (iii) sequential control (supervisory operation
10 of logic and analog control structures). In addition, every process control system must address batch (or recipe based) manufacturing with the related topics of lot tracking and validation.

The illustrated control system 10 views these styles as particular visualizations of the process control functionality. This means that these three types of control do not exist as objects. Rather,
15 the control system receives a class definition of a process control object (PCO) which it instantiates and executes. It is the responsibility of the control editor to create the correct representation of the control structure while also representing that control structure in one of the industry standard formats, i.e., ladder-logic, function blocks, or structured text, or PCOs directly

20 Sequential Function Charts are a meta-language used to control the execution of the other three languages. In the illustrated embodiment, it serves as a wrapper class for the other classes. The source code for a control scheme and the critical values for its PCO attributes, i.e., its checkpoint file, are held in the non-volatile memory of the Native device.

25 5.4.2 Process Control Objects

5.4.2.1 PCO Attributes

The fundamental control object is a PCO attribute. As noted above, these are provided in two types: variables and parameters. An example of a variable attribute is a measurement in a PID block. In the illustrated system, a PCO attribute is an object in its own right and it contains
30 certain vital information when it is retrieved. This information is "atomic", i.e., all information is obtained with an attribute, not just its "value". The minimal set of information from a PCO

variable attribute is value, data type, quality information time tag in milliseconds, and alarm status. It is this object that is transferred from one PCO to another to implement the transfer of information needed in an actual control scheme.

Parameter attributes bundle less information than a variable attribute. The typical example of a parameter attribute is the proportional band of a PID block. In the illustrated control system 10, this attribute has less supporting information than a variable attribute, e.g., it would not have an engineering unit range.

5.4.2.2 PCO Blocks

PCO attributes and methods to operate on those attributes are combined into PCO Blocks. These blocks are analogous to the control blocks in other systems, e.g., I/A Series FoxBlocks, IEC 1131-3 Function Blocks, and Foxboro Microspec Blocks. However, because they are implemented in Java, they can be executed on any microprocessor running any OS that supports a JVM. In addition, they are designed to occupy only the amount of RAM required for the selected options and to be on-line replaceable.

PCO blocks capture and encapsulate knowledge of regulatory control algorithms needed to by continuous processes. The PCO user block can be used to integrate particular user-defined control algorithms into a PCO control scheme. It is the PCO user block that is used to implement control strategies that are defined using one of the IEC 1131-3 control languages.

5.4.2.3 PCO Composites

PCO Composites are implementations of a control strategy. They consist of PCO blocks and other PCO composites linked in such a manner as to provide the control mechanisms needed for a particular process. It is the PCO composite that captures application level control expertise, e.g., complex control structures for major pieces of equipment or even the simpler idioms of cascade control.

5.4.3 PCO Execution Environment

The PCO execution environment is the software in a native control device that supports the execution of PCOs. This software includes the JVM and any other required code, such as a class loader that loads stand-alone applications for each composite, a class loader that loads an applet for each composite, a block processor that loads Java byte codes for a composite and executes each composite in turn, and a block processor that loads Java byte codes for PCO Blocks and executes each blocks in turn.

5.5 Communications

5.5.1 Requirements

5.5.1.1 Communication Model Definition

Several types of communication are provided in the illustrated control system 10. These types include data transfer, message transfer between application programs, object movement (downloads), class file (code) transfers, and events (unsolicited messages).

The communications mechanism is redundant, i.e., the failure of one station in the system does not prevent communication (new or established) between any two other stations. It also minimizes the impact on uninvolved stations when the service is used by its clients, e.g., a message is not sent to a network segment if it is not going to be used there. The communications mechanism, moreover, provides an encapsulated implementation that can be replaced "under the hood" as new technologies become available. Also, it allows the illustrated control system 10 to be configured for the appropriate communication resources, e.g., the number of outbound connections. Moreover, it provides performance statistics, e.g., high, low, average transfer times between two stations, bytes per seconds between two stations, etc., for viewing in the system monitor. Still further, it automatically reconnects to an object if it is relocated on or removed from/restored to the network.

Data transfer in the illustrated control system 10 is provided by a mechanism that permits clients to subscribe to updates in a object. It also provides a mechanism for one-shot get/set access to an object. A client of the data transfer facility supplies the control system with a (i) number of objects that it needs, (ii) conditions under which it wants updates: change delta, maximum time between updates, or both, (iii) be capable of retrieving the values all at once (a snapshot), and

(iv) be capable of retrieving the values as series of changes with the option to block, for a user specified amount of time, while waiting for a change. Client publication (write lists) are not supported in the illustrated embodiment. Target devices subscribe to the data in the source device.

With respect to client one-shots (get), the client of the data transfer facility supplies the system with a number of objects that it needs and retrieve the values all at once (a snapshot). Conversely, with respect to client one-shots (set), the client supplies the system with a number of objects that it wishes to change, and set any part of the object if it has access. The parts referred to are the supporting information in a PCO attribute as well as the value part.

5.5.1.1.1 Message Transfer

5.5.1.1.1.1 General

While the message transfer mechanism is a general case of the data transfer mechanism, its requirements are given separately. The client of the message transfer facility must support private conversations between applications using a reliable communications mechanism and support by subscription publication of information from one server to many clients.

5.5.1.1.1.2 Private Conversations

The private conversation mode allows applications to find each other by their application name: the client application is not required to know the station name or IP address of the server application. It also allow applications to send records (messages) of arbitrary size: the private conversation mode guarantees that the message boundary is recognized and that order is preserved. Moreover, it allows both the sending and receiving process to determine if the connection is lost. Still further, it supports transaction based message transfer, i.e., the sender does not get an acknowledgement until the receiver says that the message is safely stored.

5.5.1.1.1.3 Subscriptions

The subscription facility supports senders and listeners. A sender is a process that has data that may be of interest to one or more listeners. The subscription mode supports this type of operation by allowing a "sending" process to register that it will produce messages of a particular type,

allowing a "listening" process to register to receive all messages of a particular type, supporting transaction based subscription if the client so requests (this request would have no impact on the sender), and supporting transaction based publishing by a server if the server requests it (this request would have no impact on the receiver). If no listeners are registered, the subscription facility defines how the message is handled.

5.5.1.2 Security

The native communications system is linked to the native security system to ensure that unauthorized users are unable to access objects. The illustrated control system 10 does not arbitrate between users of a resource. The prevention of multiple access to an object is the responsibility of the object. The breaking of an application secured link by an operator is not a function of the illustrated control system 10.

5.5.1.3 Object Location Services

Object are located by name. To this end, the object location service supports the location of named objects; supports hierarchical object names; allows rule based specification of the name delimiting character; locates an object based on a "longest fit" because (a) not all parts of an object name are globally known and (b) not all parts of an object are in the same physical location; supports the implementation of naming scopes, i.e., limiting the visibility of names; supports the use of a name search path so that relative names can be located; is redundant; supports locating many types of objects, e.g., server processes, PCO attributes, and application programs; and supports the addition of related information about the object in the name directory, e.g., object type, short-cut information to be used by the process that owns the object, object type/implemented interface, and information. With respect to the last item, the object location services permit a client to request the name of a service that supports a specific interface. This allows new interfaces to be added to a service without breaking an old one.

5.5.1.4 Networks

The illustrated control system operates over any IP based network 48. By way of non-limiting example, the illustrated control system 10 operates over wide area networks supplied by the customer, installed plant LANs supplied by the customer, dedicated LANs supplied either by the

customer or the vendor, redundant LANs supplied by the vendor. This operation is transparent to all applications using the system 10. While it is recognized that performance are degraded over low bandwidth (phone lines) and high latency (geosynchronous satellites) networks, the system 10 optimizes as best it can to minimize the impact of different network types. In short, the illustrated control system 10 does not assume that it "owns" the network.

5.5.1.5 Quality of Service

5.5.1.5.1 Performance

Assume that the illustrated control system 10 is operating in support of a PCO in a dedicated control station (not a transmitter) or in a PC, it supports at least 10,000 object value gets/sets per second if the object is in the local station; supports at least 100 get/set per second requests from a client if the object is in a remote station; supports at least 50 get/set requests per second made to a server from a remote client; is able to locate 3000 objects per second; is able to detect a communication failure in no more than 1000 milliseconds; ensures that the time to move data from a server station to a client station (API call to API call) is less than 100 milliseconds in a normally configured network; allows a server station to move 6000 values out of its box every second; allows a client station to accept at least 2000 values every second; allows an unlimited (but, configured) number of names to be defined in the system; and supports the registration of 50,000 globally known names without requiring a reconfiguration. All server stations are able to support at least 30 client stations for continuous data feed without requiring a reconfiguration. All client stations support at least 30 server stations for continuous data feeds without requiring a reconfiguration.

5.5.1.6 APIs

Within the illustrated control system 10 environment an API are provided for all of the above services. This API are delivered as a Java Bean so that the underlying communications mechanism may be replaced without breaking all of the applications that use the service.

5.5.1.7 Maintenance and Upgrades of Messaging Services

The illustrated control system 10 defines and enforces a mechanism for supporting inter-operability of messages from stations at different versions. A mechanism like interface

inheritance is provided, i.e., a message carries in it enough "type" information that the receiving application knows how to "decode" the message even if the receiving application is at a lower level. This adds fields to messages if necessary, but not remove/replace old fields.

5.5.1.8 Integration of non-Native Based Systems

The illustrated control system 10 supports integration of non-Java applications using appropriate technologies, e.g., CORBA, COM/DCOM/ActiveX, and emulation of APIs. Physical gateways are permitted, but these gateways provide only a minimal amount of configuration.

5.5.1.9 Object Location Service Implementation

5.5.1.9.1 The All Java Approach

5.5.1.9.1.1 Description

JINI and JavaSpaces technologies permit object location within the illustrated system. JINI technology allows native devices to register names in a globally available database. JINI clients can find this database and perform searches against it. JavaSpaces allow a device to register important information in a globally available bulletin board. JavaSpaces provide four simple services: posting, removal, query by example, and notification. The notification feature informs prospective users of a class of objects when such objects are added and removed from the JavaSpace. A JavaSpace handles services like system monitoring.

5.5.1.9.2 The Enhanced Object Manager Approach

5.5.1.9.2.1 Description

As an alternative, an enhanced object manager (EOM) approach to address location and connection of data clients and servers can be utilized as follows:

1) A LDAP compliant service are created and used to store pathnames and related information, e.g., IP addresses, Port Numbers, internal indexes, etc.
2) The EOM API provides get/set and read/write support similar to that provided by the OM today.
3) When provided a name, the EOM looks in the local copy of the LDAP database.

4) If the name is found and the data is local, the request is passed to the correct data provider (the EOM, the CIO database, or the AOS database) who returns the value and other data. The EOM handles sending updates on change driven data.
5) If the name is found and not local, the EOM sends the request to the remote EOM which then makes a local query. The EOM handles sending updates on change driven data.
6) If the name is not found locally, the location request is passed to the master LDAP database. If it is found there, the address information is passed back to the local LDAP database (which is updated) and steps 4) and 5) are repeated.
7) If the name is not found remotely, the request is posted for a period of time. If it is resolved in that period, the requesting station is notified. If it is not resolved, the master LDAP database drops the name and notifies the requesting station.

5.6 Critical Applications

5.6.1 Time Synchronization

SNTP (Simple Network Time Protocol) is used to keep operator stations and similar non-process data producers in sync. This protocol is as accurate as 1 ms in a carefully controlled environment, but 50 ms is more common on shared networks. Controllers requiring high accuracy are placed on tightly controlled networks or equipped with an interrupt used to coordinate time updates. In either case, a master timekeeper tied to a well known reliable data source - GPS clock or broadcasted time from the national atomic clocks. This master time keeper generates the interrupt and SNTP. It is built with a highly reliable internal clock so that loss of time feed does not result in significant drift within the Mean Time to Repair (MTTR).

5.6.2 Alarm and Message Management

Alarm and message manage requires that all process alarming to be based on the top-level composite's attributes; that acknowledgments be handled at the composite level; that PCOs, system monitors, and the native human interface a reliable message protocol to send the message to a message management system (MMS); that the MMS is used by end-users to view the current alarm state and the alarm history; and that the MMS use the plant historian to permanently store and archive messages.

To meet the control market's requirement for a message management system, the illustrated control system 10 provides a message management support service (MMSS) in the native devices that are used to deliver messages to a Message Manager. The service operates as follows.

When a client of the MMSS connects to the MMSS server, the MMSS server send any messages in its internal alarm queue that are older than the time of the connection to the client; transmits the alarm state at the time of the connection to the client, and transmit any new messages subsequent to the connection. If the connection to the MMSS client fails, there is no impact on the operation of the control station or the control network; minor delays associated with detecting the loss of connection are generally viewed as acceptable. The MMSS support at least two (2) and, preferably, four simultaneous clients.

The alarm state of a PCO attribute is defined to include block attribute identification (full PATH attribute name, at least); the bad I/O status of the block for I/O attributes; an indication of the alarm status for each alarm type defined for the attribute; the priority, criticality, and acknowledgment status of the attribute; the on/off status of the attribute, i.e., is it updating; the status of the alarm inhibit and alarm disable or alarm option parameters, and the time tag of each alarm event stored within the block, e.g., the into alarm time, the acknowledged time, and the return to normal time. This information is recovered from all currently active alarms.

5.6.2.1 Configuration

5.6.2.1.1 Process Alarms

When a composite is configured, the application engineer specifies which attributes of the PCOs within the composite are to be annunciated. When a composite is placed on line, it notifies the object location service that it exists. A message management service are notified of the new composite and arrange for notification from the device.

5.6.2.1.2 System Alarms and Messages

No user configuration is required to get these alarms into the MMS. The sources of such messages arranges for the appropriate notification.

5.6.2.1.3 Native Operator Actions

No user configuration is required to get these alarms into the MMS. The sources of such messages arranges for the appropriate notification.

5.6.2.2 Native Message Management Service

The MMS are centralized and redundant. It provides a complete alarm state for the plant at any time including startup. It retains a message history for as long as configured through the use of the plant historian. Users view the alarms through to the Alarm Manager.

6 Native Hardware SubSystems

6.1 Field Devices

6.1.1 Overview

The native field devices are coupled to the network 48 via an IP network, preferably, Ethernet and, still more preferably, powered Ethernet. Powered Ethernet delivers the power, along with conventional Ethernet data. To this end, the native field devices use one of two wiring schemes, with each scheme supplying power to the field device: (a) standard four wire 10Base-T wiring, or (b) industry standard two-wires used for 4-20ma based transmitters. The devices also provide a single Ethernet interface (or redundancy, where desired in the cabling or the field device); and connect to a non-redundant powered hub with the following features: (a) non-redundant connections to the IFDs; and (b) support for redundant connection to a plant-wide network. Where used, the powered Ethernet implementation leaves the physical and logical Ethernet interfaces in place. This means that except for adding a new wiring type, the physical layer, the data link layer, the use of CSMA/CD, and 10Base-T connectors are preserved.

The illustrated embodiment utilizes powered Ethernet hubs to provide communications and power to its intelligent field devices. These IFDs may be transmitters, actuators, Networked I/O, or PCO platforms. The powered hubs are preferably, stackable, DIN rail-mountable, support connection to a redundant network, and support both 10 Mbps and 100 Mbps Ethernet.

A further understanding of the operated of the powered Ethernet network and of the circuitry used within the native field and control devices to draw power therefrom may be attained by reference to United States Patent Application Serial No. 09/444,827, filed November 22, 1999, entitled POWERED ETHERNET FOR INSTRUMENTATION AND CONTROL, and by reference to the corresponding PCT Patent Application US00/10013, filed April 14, 2000, the teachings of which are incorporated herein by reference, and a copy of which is attached as an appendix hereto.

On the software side, specific implementations of the native field devices support web based configuration, as described above. The support is supplied by including an embedded web server with a selection of pages used for configuration and maintenance.

6.1.2 Transmitters

Intelligent transmitters use the IEEE 1451 standard to communicate to their sensor(s). This facilitates use of the same module in many types of transmitters from potentially many vendors and allows the transmitter to support up to 255, perhaps externally mounted, sensing devices.

6.1.3 Positioners

The native intelligent positioners also support a powered Ethernet interface like the one used by the native intelligent transmitters. Figure 6 shows a preferred native intelligent positioner implementation. The approach standardize interfaces and use a PCO to control the positioner as evident in the drawing.

6.2 Summary of Hardware/Software Features by Station Type

A summary of the features of native control devices follows. Blank cells indicate that the feature is not present in the indicated device.

Features	Xmitter	Positioner	Controller	Integrator	PC based Operator Consoles	Native Workstation	PC Based Web Server	Solid-state Web Server	SS DHCP Server	SS bulletin board
Low power, high performance CPU	Yes	Yes				Yes				
Top performance CPU			Yes	Yes	Yes	Yes	Yes	Yes		
RAM	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ROM	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Flash	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes
Sensor I/F (IEEE 1451)	Yes	Yes								
Powered Ethernet (2 wire)	Yes	Yes								
Serial I/F (RS-232/485/422/423)	Yes	Yes		Yes						
Ethernet (10Base-T)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Features	Xmitter	Positioner	Controller	Integrator	PC based Operator Consoles	Native Workstation	PC Based Web Server	Solid-state Web Server	SS DHCP Server	SS bulletin board
Wireless Ethernet						Yes		Yes		
the TCP/IP suite	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DHCP Server							Yes	Yes	Yes	Yes
Full OS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Java Virtual Machine	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Web Server	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes
Device Configurator	Yes	Yes	Yes	Yes					Yes	Yes
Annunciator Keyboard					Yes	Yes				
Touchscreen					Yes	Yes				
Mouse/Trackball					Yes	Yes				
Display					Yes	Yes	Yes	Yes		
Hard drive					Yes		Yes			
Solid State Bulk Storage						Yes		Yes		
System Monitor							Yes	Yes		
Process Control	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		

Features	Xmitter	Positioner	Controller	Integrator	PC based Operator Consoles	Native Workstation	PC Based Web Server	Solid-state Web Server	SS DHCP Server	SS bulletin board
Objects (PCOs)										
System Management Objects	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Omnibus Messaging Services	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		
Profibus PA Support	Opt.	Opt.		Yes						
Profibus DP Support	Opt.	Opt.		Yes						
HART Support	Opt.	Opt.		Yes						
FoxComm Support	Opt.	Opt.		Yes						
Fieldbus HSE Support	Opt.	Opt.		Yes						
Fieldbus H1 Support	Opt.	Opt.		Yes						
DeviceNet	Opt.	Opt.		Yes						

Features	Xmitter	Positioner	Controller	Integrator	PC based Operator Consoles	Native Workstation	PC Based Web Server	Solid-state Web Server	SS DHCP Server	SS bulletin board
LONworks	Opt.	Opt.		Yes						
Modbus				Yes						
Modbus+				Yes						
ABDH				Yes						
I/A Series Nodebus				Yes						
I/A Series Fieldbus (FoxComm)	Opt.	Opt.		Yes						
Application Development Environment							Yes			
Historian							Yes			
Time Synchronization							Yes	Yes		
PIMS							Yes			
Message Management							Yes			

Described above are apparatus and methods that achieve the goals of the invention. It will be appreciated that the embodiments described herein are examples and that other embodiment incorporating changes thereto also fall within the scope of invention. Thus, by way of example, it will be appreciated that the invention can be implemented using a virtual machine environment other than JVM and using bytecode other than Java bytecode. By way of further example, it will be appreciated that the apparatus and methods taught herein can be applied to a range of control application, in addition to process control.

In view of the foregoing, what I claim is

Appendix (from PCT Patent Application US00/10013, filed April 14, 2000)

Appendix to Patent Application for

5

METHODS AND APPARATUS FOR CONTROL USING CONTROL DEVICES THAT PROVIDE A VIRTUAL MACHINE ENVIRONMENT AND THAT COMMUNICATE VIA AN IP NETWORK

Appendix (from PCT Patent Application US00/10013, filed April 14, 2000)

POWERED ETHERNET FOR INSTRUMENTATION AND CONTROL

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority from U.S. Provisional Application Nos. 60/129,541 and 60/157,110, which were filed, respectively, on April 6, 1999, and October 4, 1999, both of which are incorporated by reference.

TECHNICAL FIELD

The invention relates to interconnecting devices for instrumentation and control.

BACKGROUND

Control systems often include intelligent field devices having digital processors. Examples of such devices include valve controllers and transmitters, such as those associated with, for example, temperature sensors, pressure sensors, and flow meters. Another type of intelligent field device is a field mounted process controller that sends control signals to, for example, a valve controller, based on process information received from one or more transmitters.

The functionality of different field devices may be incorporated into a single module. For example, a process controller, a valve controller, and a temperature sensor/transmitter may be incorporated into a single module. However, even with such consolidation of function, control systems typically require some mechanism for transmitting signals between devices, and for transmitting signals from devices to a central control station.

Due to cost constraints, connections between devices, and between the devices and a control station, are often provided using a single pair of wires. In many instances, this same pair of wires is used to provide power to the devices.

Many proprietary protocols have been developed for powering field devices and transmitting signals between the devices using a single pair of wires. Two such protocols are the HART protocol and the FOXCOMM protocol.

Appendix (from PCT Patent Application US00/10013, filed April 14, 2000)

The HART protocol specifies provision of a 4-20 mA analog control signal and a 1200 baud bidirectional digital data link on a pair of wires connected to a device. The 4-20 mA signal is used both to power the device and to provide a control signal to or from the device. The HART protocol achieves simultaneous analog and digital transmission by using a frequency shift keying (FSK) technique to overlay a bidirectional digital signal on the analog control signal.

The FOXCOMM protocol also overlays a bidirectional digital data signal on a 4-20 mA analog control and power signal. Different versions of the protocol use either 600 baud or 4800 baud digital signals. Yet another version uses the analog signal only for power delivery and transmits all information using digital signals.

SUMMARY

In one general aspect, the invention features providing a powered Ethernet connection between two devices connected by an Ethernet connection that provides data communication between the devices. Electrical power is applied on to the Ethernet connection at the first device. At the second device, the electrical power is extracted and used to power the second device.

Embodiments may include one or more of the following features. For example, the Ethernet connection may include two pairs of wires, with a first pair of wires being used to transmit data from the first device to the second device and a second pair of wires being used to transmit data from the second device to the first device. In this case, applying electrical power on to the Ethernet connection at the first device may include applying a DC voltage across the two pairs of wires by coupling a first potential to the first pair of wires and a second potential to the second pair of wires, with the DC voltage being defined as a difference between the two potentials. For example, when each pair of wires is connected to a corresponding transformer at the first device, and each transformer includes a center-tapped primary winding, coupling potentials to the pairs of wires may include applying the DC voltage between the center taps of the primary windings of the two transformers. As an alternative, a center-tapped inductor may be connected across each pair of wires at the first device, and potentials may be coupled to the pairs of wires by applying the DC voltage between the center taps of the inductors. Electrical power may be extracted at the second

device through center-tapped windings of transformers of the second device. or through center-tapped inductors coupled across the pairs of wires at the second device.

The Ethernet connection also may be implemented using a single pair of wires to transmit data from the first device to the second device, to transmit data from the second device to the first device, and to provide power from the first device to the second device. In this case, applying electrical power on to the Ethernet connection at the first device may include applying a DC voltage across the pair of wires by coupling a first potential to the first wire through an inductor and coupling a second potential to the second wire through an inductor, with the DC voltage being defined as a difference between the two potentials.

Each device may be, for example, a process controller, a field device, or an Ethernet hub. A field device may be operable to sense a process condition and to transmit information about the sensed process condition using the Ethernet connection, or to control a process condition in response to a command received through the Ethernet connection.

In another general aspect, the invention features connecting a four-terminal Ethernet connection to a two-wire Ethernet connection. A device having a four-terminal Ethernet connection, the connection including a first pair of terminals for transmitting data away from the device and a second pair of terminals for transmitting data to the device is connected to a two-wire Ethernet connection for transmitting data to and from the device by a switched connection. The switched connection operating in a first mode in which the two-wire Ethernet connection is connected to the first pair of terminals and a second mode in which the two-wire Ethernet connection is connected to the second pair of terminals. The switched connection is initially set to operate in the first mode, and the two-wire Ethernet connection is monitored for data being transmitted to the device. The switched connection is set to operate in the second mode upon detection of data being transmitted to the device.

Power may be injected on to the two-wire Ethernet connection at the device for use in powering another device connected to the two-wire Ethernet connection. In addition, an impedance presented to the two-wire Ethernet connection may be changed from a first impedance when the switched connection is operating in the first mode to a second impedance when the switched connection is operating in the second mode.

In another general aspect, the invention features a field device having a powered Ethernet connection. The field device includes communications circuitry operable to

communicate data over an Ethernet connection using an Ethernet protocol, power circuitry operable to extract operating power from the Ethernet connection, and process circuitry operable to sense a process condition and to transmit information about the sensed process condition using the Ethernet connection, or to control a process condition in response to a command received through the Ethernet connection.

Examples of such a field device include a device providing a temperature or pressure sensor, a flow meter, and a valve controller. The communications circuitry and the power circuitry may be operable to interface with an Ethernet connection including only a single pair of wires.

Other features and advantages will be apparent from the following description, including the drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Figs. 1-3 are block diagrams of systems using powered Ethernet connections.

Figs. 4 and 5 are block diagrams of systems for providing power to a field device over a four-wire Ethernet connection.

Fig. 6 is a circuit diagram of a four-wire circuit.

Fig. 7 is a block diagram of a system for providing power and emulating a four-wire Ethernet connection using only a single pair of wires.

Figs. 8A-8F are block diagrams of different operating states of the system of Fig. 7.

Fig. 9 is a circuit diagram showing connections of a hub using two-wire powered Ethernet.

Fig. 10 is a circuit diagram of a line circuit of the circuit of Fig. 9.

Figs. 11-17 are circuit diagrams of elements of a two-wire circuit.

DETAILED DESCRIPTION

Referring to Fig. 1, a control system 100 includes a controller 105 that powers and controls a remote field device 110 over a powered Ethernet connection 115. Thus, the system 100 applies the well-known and well-established Ethernet protocol to an instrumentation and control system.

Appendix (from PCT Patent Application US00/10013, filed April 14, 2000)

The powered Ethernet connection 115 may be provided by modifying connections to existing wiring of a system employing a proprietary protocol. Thus, a powered Ethernet system may be produced by retrofitting an existing system.

Protocols such as the Ethernet protocol are often used to communicate between devices, such as components of a computer network. Ethernet uses a bus or star topology and supports data transfer rates of 10 Mbps. Ethernet handles simultaneous demands using a technique known as carrier sense multi-access with collision detection ("CSMA/CD"). The Ethernet specification served as the basis for the IEEE 802.3 communication standard, which specifies the physical and lower software layers. The IEEE 802.3 standard controls the implementation of the basic Ethernet hardware technology. In specific terms, it controls the media access control sublayer and the physical layer functions for a bus-structured local area network that uses CSMA/CD as an access protocol.

One of several adaptations of the Ethernet (IEEE 802.3) standard for Local Area Networks (LANs) is 10Base-T. The 10Base-T standard (also called Twisted Pair Ethernet) uses a twisted-pair cable with maximum lengths of 100 meters. The cable is thinner and more flexible than the coaxial cable used for the 10Base-2 or 10Base-5 standards. Cables in the 10Base-T system connect with RJ-45 connectors. A star topology is common, with 12 or more computers connected directly to a hub or a concentrator. The 10Base-T system operates at 10 Mbps and uses baseband transmission methods.

A networking standard that supports data transfer rates up to 100 Mbps (100 megabits per second), called 100BASE-T, is based on the older Ethernet standard. Because it is 10 times faster than Ethernet, it is often referred to as Fast Ethernet. Officially, the 100BASE-T standard is IEEE 802.3u.

Information is communicated on the Ethernet using electrical signals. The standard Ethernet protocol specifies the physical layer characteristics of these signals. This information is converted to digital information by the Ethernet devices.

The Ethernet protocol typically employs a first pair of wires for transmission in one direction and a second pair of wires for transmission in the opposite direction. The pairs of wires generally are not used to power the devices. Associated with the Ethernet protocol is a well-developed collection of standard hardware and software components that may be used to implement communications between devices.

Appendix (from PCT Patent Application US00/10013, filed April 14, 2000)

The powered Ethernet connection 115 provides a connection for powering and communicating with remote field devices using standard, or slightly modified, Ethernet hardware and software. The connection 115 may employ either one or two pairs of wires.

Figs. 2 and 3 illustrate examples of other system topologies. As shown in Fig. 2, a system 200 includes a controller 205 connected to a collection of field devices 210 by a powered Ethernet connection 215 in a bus configuration. The connection 215 may employ either one or two pairs of wires.

Similarly, Fig. 3 shows a system 300 that includes a controller 305 connected to a hub 310 by a powered Ethernet connection 315. The hub 310 is then connected to a collection of field devices 320 by powered Ethernet connections 325 in a star configuration.

Each of the connections 315, 325 may employ either one or two pairs of wires, and different connections may employ different numbers of pairs. For example, the connection 315, which is typically substantially longer than the connections 325, may employ a single pair of wires, while one or more of the connections 325 employs two pairs of wires. Exemplary single-pair (two-wire) and double-pair (four-wire) systems are discussed in detail below.

Four-Wire System

Referring to Fig. 4, a system 400 employs standard Ethernet boards 405 at both a process controller 410 and a remote field device 415. The boards communicate via two pairs of wires, with the first pair 420 providing communications from the process controller to the field device, and the second pair 425 providing communications from the field device to the process controller.

Each Ethernet board includes a pair of transformers 430, with each transformer being associated with a pair of wires. At the process controller, a 24 Volt DC power signal from a power supply 435 is injected to central taps 440 of the primary coils of the transformers. The positive (+V) lead 445 of the DC power signal is provided at the central tap of the primary coil of one of the transformers. The ground (GND) lead 450 of the DC power signal is provided at the central tap of the primary coil of the other transformer.

At the field device, the central tap 440 of the primary coil of one of the transformers 430 serves as device ground 455. The central tap 440 of the primary coil of the other

transformer 430 provides a device operating power signal 460. The signal 460 and ground 455 are provided to a DC-to-DC converter 465 that reduces and isolates the DC voltage to provide operating power for the device. For optimum power transfer, the coil of the DC-to-DC converter can be impedance matched to the resistance of the wire pairs. The output of the DC-to-DC converter is provided to a voltage regulator circuit 470 that provides power to the Ethernet board and the field device. The process controller also may include a DC-to-DC converter 465 and a voltage regulator circuit 470 for use in powering the Ethernet board of the process controller. These elements are unnecessary when the process controller has a separate power supply.

Only one remote field device 415 is shown in Fig. 4. However, as illustrated in Figs. 2 and 3, nothing precludes having more devices connected at the far end of the Ethernet wire pairs. The only requirements are that the loads of the devices must be designed so that the wire pairs are properly terminated and that the total power delivered is sufficient to drive all of the devices. Since only one remote device is active at a time, the power requirement will be the sum of the requirements of the one active device and N - 1 idle devices, where N is the number of devices connected to the wire pair.

The transformers include primary coils connected to the twisted pairs and secondary coils connected to devices on the Ethernet boards. The secondary coils of the transformers may be constructed of bi-filar coils to reduce the required size of the transformers.

Referring to Fig. 5, for transformers that cannot carry the required DC current or don't have center taps in their primary coils, the power can be applied via center-tapped inductors 500 connected across the secondary coils of the transformers. The DC current is isolated from the transformers by capacitors 505. Each inductor must be large enough that it does not degrade the termination impedance of its corresponding wire pair.

Fig. 6 illustrates a fairly simple prototype circuit 600 for powering a four-wire connection 605. At the process controller end 610, center-tapped inductors 615 are used to direct power on to the connection 605. At the field device end 620, center-tapped inductors 625 are used to remove power from the connection 605. With the circuit shown in Fig. 6, a 24 volt signal was applied to the connection 605 at the process controller side and about 100 mA was drained through a resistor 630 on the field device end, resulting in power consumption of 2.5 watts. The two ends were implemented using NIC cards 635 mounted in

personal computers. No attempt was made to actually power any part of the NIC cards. The loop was about six feet long, which is much shorter than many contemplated implementations. The prototype circuit confirmed that power can be successfully injected into an Ethernet connection without jeopardizing the communications capability of the connection.

Two-Wire System

Referring to Fig. 7, a system 700 having a two-wire connection may be used to emulate a four-wire, powered Ethernet connection. This approach is particularly useful in retrofitting Ethernet to existing systems that use only a single pair of wires to provide power and communications between a process controller and a remote field device. In particular, this approach may be used to make Ethernet connections without upgrading or replacing existing wires.

The system 700 includes a hub port 705 connected by a single pair of wires 710 to a remote field device 715. Each connection includes the hub port 705, a switch 720 connected between the port and the hub end of the single pair of wires 710, an activity detector 725 connected to the switch 720, and a switch 730 connected between the other end of the pair of wires and the remote field device. Each switch functions as a double pole, double throw switch, and may be implemented with a driver and receiver that can be enabled or disabled by means of a chip select lead. The switches control their associated devices to be in reception states or transmission states.

Referring to Fig. 8A, the remote field device 715 and an associated hub port 705 operate in an idle mode when no traffic is present on the wire connection between them. The hub switch 720 is switched to the transmit position and the remote field switch 730 is switched to the receive position. In this mode, the remote field device can determine if the network is clear.

Referring to Fig. 8B, when the field device 715 is ready to send data, the field device 715 switches the switch 730 to the transmit position under control of the Ethernet remote physical layer. This connects the field device to the hub activity detector 725. The activity detector 725 detects the first part of the output signal of the remote device 715.

Appendix (from PCT Patent Application US00/10013, filed April 14, 2000)

Referring to Fig. 8C, once the hub's activity detector 725 detects activity, it switches the switch 720 to the receive position to connect the transmitting field device to the transmit pair of the hub. There may be several bits of transmission lost during this transition. However, this causes no problem since the first 64 bits of the Ethernet packet are only used for synchronizing clocks.

Referring to Fig. 8D, once the hub starts receiving data, the hub generates a signal to the other remote field devices to notify them that the system is busy so that they will not start to send data of their own.

Referring to Fig. 8E, the hub may detect a collision when it receives a signal on its receive pair from another remote field device connected to it. It must then signal to the remote field devices that a collision has occurred. This can be done in several ways. One simple way is to reverse the polarity of the DC power supplied to the field devices. The field devices may be configured to automatically switch their switches to the receive positions in response to this polarity reversal, as shown in Fig. 8F. The transmitting field devices will then see the jamming signal on their receive inputs, will stop transmitting, and will initiate a backout sequence to resolve the collision.

Hub

Referring to Fig. 9, an eight-port, standard Ethernet hub 900 includes ports 905, each of which includes a two-wire transmit connection 910 that receives data transmitted by a remote device 715 and a two-wire receive connection 915 that sends data to be received by the remote device 715. A two-wire powered Ethernet connection is provided by inserting a line circuit 920 between a hub port 905 and a two-wire connection 710 to a remote device 715. This permits the use of standard Ethernet hubs to provide powered connections to remote devices over only a single pair of wires.

Referring to Fig. 10, each hub port line circuit 920 includes a first pair of terminals 1000 connected to the two-wire connection 710, a second pair of terminals 1005 connected to the transmit connection 910, and a third pair of terminals 1010 connected to the receive connection 915. Power is supplied to the terminals 1000 by a 24 Volt source connected to the terminals through inductors 1015 or other high impedance devices. Capacitors 1020

serve to isolate the power from a transformer 1025, the secondary coil of which is connected through the capacitors 1020 to the terminals 1000.

A load switch 1030 is connected across the primary coil of the transformer 1025. The load switch 1030 is controlled by the hub to provide impedance matching. When the hub port is in the receive mode, the load switch 1030 connects a load impedance across the line so that the line is properly terminated. When the hub port is in the transmit mode, the load switch removes the load since a transmission driver 1035 provides the terminating impedance.

An activity detector 1040 also is connected across the primary coil of the transformer 1025. As discussed above, the activity detector 1040 detects transmissions by the remote device and notifies the hub to permit the hub to reconfigure the line circuit 920. The activity detector can be implemented in a number of ways. One simple implementation rectifies the signal produced by the primary coil of the transformer 1025 and compares the rectified signal to a threshold that indicates that a signal is present.

A receiver 1045 is connected across the primary coil of the transformer 1025. Outputs of the receiver 1045 are connected to the pair of terminals 1005. The receiver 1045 is controlled by a signal (TCE) from the hub.

Outputs of the transmission driver 1035 are connected through resistors 1050 to the primary coil of the transformer 1025. The resistors 1050 may be used to provide a 100 ohm driving impedance. Inputs of the transmission driver 1035 are connected to the pair of terminals 1010. The transmission driver 1035 is controlled by a signal (RCE) from the hub.

Two-Wire System - Details

As discussed above with reference to Fig. 7, a powered, two-wire 10 MHz half duplex Ethernet communications system may be used to transmit Ethernet packets as well as DC power to remote field devices. This permits existing 4-20 mA loop-powered remote field devices to be upgraded to intelligent field devices using their existing 4-20 mA twisted pair current loop cabling. This eliminates the substantial expense of replacing existing current loop wiring to upgrade the customer's cable to UTP Category 5 Ethernet wire.

To understand how the two-wire powered Ethernet communication system operates, it is useful to understand how conventional 10 BaseT Ethernet operates. To this end, three

conventional Ethernet configurations are considered below: (1) point-to-point communication, (2) multi-point communication through an Ethernet hub, and (3) multi-point communication through an Ethernet switch.

Point-to-Point

The first configuration examined was a two-wire powered Ethernet point-to-point connection that involved two NIC cards connected together via a cable that was cut in half so that the two pairs of the cable could be connected together. With the two pairs shorted together, the NIC cards were put in the full-duplex mode so that a card that received its own signal would not interpret the signal as a collision and shut down. The two cards communicated with each other, by "pinging" each other continuously. Pinging is simply sending a message with an address of a device. If the device is up and connected, it sends an acknowledgement.

This configuration used two standard NIC cards from SMC installed on two standard personal computers running Windows NT. The PHY chip on the SMC NIC cards is the Altima AH101 single port chip. As noted above, the TX output of each NIC card was connected to its RX input to reduce the channel from four wires down to two. Both cards were set to 10MHz, full-duplex operation in Windows NT using the NIC cards' standard driver software. This setting did not allow the two NIC cards to perform auto-negotiation, which would have caused communications to fail. In addition, the cable was a short run (approximately 10 ft.) of Category 5 unshielded twisted pair.

No hardware changes were made to the NIC cards themselves. The two wire connection of the TX pair to the RX pair was made external to the NIC card on a small adapter board. In addition, no power was sent during initial testing.

DC power was later introduced into this configuration using several discrete analog components. These modifications were added to the external adapter board. Additions made to the adapter boards to superimpose DC power into the two-wire cable included using inductors to couple DC power on to the two-wire composite pair, while adding capacitors to AC couple the high speed Ethernet signals to the wires. Ten volts of DC power was applied to one adapter board and was sent across a short run of Category 5 unshielded twisted pair to the other adapter board on which was mounted a 10 Ohm, 10 Watt power resistor. With 10

volts sent across the two-wire composite pair, a current of 100 mA was delivered to the resistor, which resulted in a total of 1 Watt being sent across the cable, simultaneous with Ethernet signal information (i.e., pinging) being sent between the two NIC cards.

Multi-Point with Hub (No Modifications to Hub)

The second configuration, which proved unsuccessful, employed a standard NIC card connected to a standard hub and back to a standard NIC card. The configuration included one NIC card, again the SMC 9432 NIC card using the Altima AH101 PHY chip, connected in two-wire mode via UTP Category 5 cable to one port of a standard low cost single board Ethernet hub. Another 9432 NIC card (in another computer) was interfaced in standard Ethernet 4 wire mode via a short run (10 ft.) of UTP Category 5 cabling to another port of the Ethernet hub. When tested, this configuration failed because the hub is always a half duplex only device.

When a hub is connected in two-wire mode to some other two-wire Ethernet device, such as a NIC card, the TX output of that particular hub port may be directly connected to the RX input of that same port. Therefore, when the hub transmits out of one of its ports in two-wire mode, the TX output data is immediately received by the corresponding RX input. When a device, such as a hub, operates in half duplex mode, it can only receive when it is not transmitting and it can only transmit when it is not receiving. That is the definition of half duplex operation. Full duplex operation, on the other hand, allows a device to transmit and receive simultaneously, which is why it needs four wires. When a half duplex device receives a packet while it is transmitting, it considers this to be a collision, which is a violation of the half duplex mode of operation. The generic response of a half duplex device to a perceived collision is to immediately terminate whatever it was transmitting when something came in on its RX input, ignore the packet it is receiving on RX, generate a "jam" signal on its TX output, and then time out for some random period of time before trying to re-transmit the packet it was sending when the collision occurred. However, when it tries to re-transmit the packet, it again immediately senses the packet on its RX input, and again goes into collision response mode. This sequence of events continues indefinitely whenever the hub tries to transmit an Ethernet packet, which means that no useful information is transmitted as long as TX is connected to RX. Therefore, without hardware or firmware

modifications, a standard half-duplex, off-the-shelf hub cannot be used for two-wire Ethernet communication in a system. This problem is avoided by using a hub that functions in full duplex mode.

Multi-Point With Switch (No Hardware Modifications Made to Switch)

A third configuration used a standard NIC card interfaced to a port of a high performance Cisco switch in two-wire mode and another NIC card interfaced to the switch via another switch port in four-wire mode with no modifications made to the switch's hardware. (Changes were made to the switch firmware via the software control console on the switch.) The NIC-card-to-Cisco switch two-wire interface failed because the switch could not properly perform a needed operational function called auto-negotiation with a two-wire cable. The problems can be avoided by directing the switch to not auto-negotiate with the NIC in two-wire mode and to disregard anything it receives on its RX input when it is transmitting out of TX on its two-wire interface.

Multi-Point with Switch (Modifications Made to Switch)

A variation on the previous configuration used a low cost, single board, ADMtech switch of simple construction, made with three major VLSI components and other low cost discrete chips and components. The switch was modified by adding one low cost HC chip and a resistor to the board. The rest of the experimental configuration was the same as that for the previous configuration. In particular, one NIC card was interfaced to the switch in two-wire mode on one switch port, while a second NIC card was interfaced to the switch via another port in four-wire mode.

This configuration worked correctly and as designed because the modification to the switch prevented the switch from paying attention to packets being immediately received on the two-wire port when the TX output was transmitting out (the usual two-wire mode transmission problem). The other reason this worked is because the switch was configured during hardware setup to auto-negotiate with the NIC before it saw the two-wire interface. The lesson learned from this configuration is that a simple, low cost switch may be a viable candidate for use in a two-wire Ethernet system configuration, if the switch can be configured, through hardware or other modifications, to ignore packets being transmitted in

each two-wire port by the corresponding RX input and, through hardware or firmware modifications, to not auto-negotiate. The hardware modification to the switch involving the 74HC125 chip and resistor solves the problem of RX inputs ignoring transmitted TX packets. The second problem is solved by adding a boot strap serial EEPROM to the board to direct the main switch controller chip, the ADMtech AL968 chip, to not auto-negotiate.

The first modification of the NIC board attempted for two-wire operation is shown in Fig. 11. The NPN transistor 1100 was turned on only when TXEN went high. This connected the two 49.9 ohm resistors 1105 to TXOP and TXON. When the TXEN lead was low, the transmit driver was disconnected from the rest of the circuit. The receive pair on the output side of the transformer 1110 was connected to the transmit pair. The transmit pair was then connected to the jack 1115 and inductors were added. This approach was unsuccessful.

The next modification approach is shown in Fig. 12. All modifications were done to an external board 1200 that was then connected in series with the loop. This approach was successful and resulted in the two NIC cards communicating. One problem with this approach was that the terminations on the transmit and receive pairs were connected in parallel, which resulted in a 50 ohm termination instead of a 100 ohm termination. This is not important on short loops. With the setup shown in Fig. 12, 10 volts were applied to one card and a 100 ohm resistor on the other. This resulted in one watt being sent over the loop.

A telephone then was connected to each loop, and one end was connected to a 20-volt power supply through two 160-ohm resistors. Users were able to talk on the Ethernet loop while it was transmitting data. The circuit is shown in Fig. 13.

Other attempts to have the cards communicate through an unmodified Altima switch were unsuccessful. Since the Altima switch uses virtually the same physical layer, AHlO4L vs. AH101L, it was determined that the problem was because the switch is smarter than the NIC cards. The switch generated a table of addresses by monitoring the traffic. Once it saw its own address as the source of an incoming message, it shut down. This problem may be solved through use of a card that cuts off the receiver when the transmitter is active. Such a card is shown in Fig. 14. In Fig. 14, resistors R3 and R4 have been added to provide a DC current component since Tl is not a bipolar switch. L3 is added so that the DC current will

not saturate the transformer. With this circuit, the only modification needed on the Altima switch is to tap into the TXEN signal and to pick up a ground.

Maximum Connection Distances

The maximum distance over which powered Ethernet can operate is a function of several variables. The distance can be limited by one of two factors, the power limitation and the signal strength limitation. The power limitation is in turn determined by the power a device needs, the resistance of the loop, and the maximum voltage available for use. To get the maximum distance, the DC impedance of the converter must equal the loop resistance. In general, the connection distance provided by the four-wire mode is twice that of the two-wire mode. Assuming that the voltage is restricted to 24 volts, that 26 gauge twisted pair wiring is used, and that the power requirement for the device is one watt, the loop and the device will each dissipate one watt (since, for maximum distance, the device resistance equals the loop resistance.) The DC current will be 83.3 mA, and the loop resistance will be 144 ohms. For two-wire operation this corresponds to a loop length of 1,450 feet. For four-wire operation, the length is 2,900 ft. This is sufficiently long that the limitation due to signal strength may be the controlling factor for most applications.

While physical layer chips advertise that they can support a distance of 100 meters, Altima states that they guarantee a minimum of 140 meters at 100 Mbit operation. The limit for 10 Mbit operation should be considerably longer. Another possibility for greater signal distance is to operate at 1 Mbit. However, this would require modifications to the switch.

Full Duplex

There are a number of problems with developing true full duplex over a single pair. It requires the development of a hybrid similar to those used in the telephone. A hybrid circuit requires a point where the output signal can be picked off uncorrupted by any input signal. The present physical layer chip does not provide such a point. The chip's output is from a current source that drives the line in parallel with a 100 ohm terminating resistor. This output circuit is shown in Fig. 15. If the output were as shown in Fig. 16, the desired signal could be picked off at points a and b. One way to provide full duplex operation is shown in Fig. 17. The 100 ohm resistor connected to the two 49.9 ohm resistors is required

for the output to be the proper shape for Manchester encoding. The added circuit essentially doubles the power used by the output driver.

Multi-drop

Providing multi-drop capability is even more difficult than full duplex operation. To prevent reflections at the point of connection, the output driver must present an infinite output impedance (that is, it must be a pure current source). Since the signal injected in the loop will travel in both directions, the driver power will be doubled. The power supply also becomes more complicated. The power supplies on the remote devices must be designed so that they draw a constant current. Otherwise, near-end devices will draw excessive current and far end devices will not be able to operate.

Other embodiments are within the scope of the following claims.

What is claimed is:

Appendix (from PCT Patent Application US00/10013, filed April 14, 2000)

1. A method of providing a powered Ethernet connection between two devices, the method comprising:

providing a first device;

providing a second device;

providing an Ethernet connection between the first device and the second device, the Ethernet connection providing data communication between the devices;

applying electrical power on to the Ethernet connection at the first device;

extracting electrical power from the Ethernet connection at the second device; and

powering the second device using the extracted electrical power.

2. The method of claim 1, wherein the Ethernet connection comprises two pairs of wires, with a first pair of wires being used to transmit data from the first device to the second device and a second pair of wires being used to transmit data from the second device to the first device.

3. The method of claim 2, wherein applying electrical power on to the Ethernet connection at the first device comprises applying a DC voltage across the two pairs of wires by coupling a first potential to the first pair of wires and a second potential to the second pair of wires, with the DC voltage being defined as a difference between the two potentials.

4. The method of claim 3, wherein:

each pair of wires is connected to a corresponding transformer at the first device,

each transformer includes a center-tapped primary winding, and

coupling potentials to the pairs of wires comprises applying the DC voltage between the center taps of the primary windings of the two transformers.

5. The method of claim 3, wherein:

a center-tapped inductor is connected across each pair of wires at the first device, and

coupling potentials to the pairs of wires comprises applying the DC voltage between the center taps of the inductors.

Appendix

6. The method of claim 3, wherein:

each pair of wires is connected to a corresponding transformer at the second device,

each transformer includes a center-tapped winding, and

extracting electrical power from the Ethernet connection at the second device comprises extracting a DC voltage existing between the center taps of the windings of the two transformers.

7. The method of claim 3, wherein:

a center-tapped inductor is connected across each pair of wires at the second device, and

extracting electrical power from the Ethernet connection at the second device comprises extracting a DC voltage existing between the center taps of the inductors.

8. The method of claim 1, wherein the Ethernet connection comprises a single pair of wires used to transmit data from the first device to the second device, to transmit data from the second device to the first device, and to provide power from the first device to the second device.

9. The method of claim 8, wherein applying electrical power on to the Ethernet connection at the first device comprises applying a DC voltage across the pair of wires by coupling a first potential to the first wire through an inductor and coupling a second potential to the second wire through an inductor, with the DC voltage being defined as a difference between the two potentials.

10. The method of claim 1, wherein the first device comprises a process controller.

11. The method of claim 10, wherein the second device comprises a field device operable to sense a process condition and to transmit information about the sensed process condition to the first device using the Ethernet connection.

Appendix

12. The method of claim 10, wherein the second device comprises a field device operable to control a process condition in response to a command sent from the first device using the Ethernet connection.

13. The method of claim 1, wherein the first device comprises an Ethernet hub.

14. The method of claim 13, wherein the second device comprises a field device operable to sense a process condition and to transmit information about the sensed process condition to the first device using the Ethernet connection.

15. The method of claim 13, wherein the second device comprises a field device operable to control a process condition in response to a command sent from the first device using the Ethernet connection.

16. The method of claim 1, wherein the second device comprises a field device operable to sense a process condition and to transmit information about the sensed process condition to the first device using the Ethernet connection.

17. The method of claim 1, wherein the second device comprises a field device operable to control a process condition in response to a command sent from the first device using the Ethernet connection.

18. A method of connecting a four-terminal Ethernet connection to a two-wire Ethernet connection, the method comprising:

providing a device having a four-terminal Ethernet connection, the connection including a first pair of terminals for transmitting data away from the device and a second pair of terminals for transmitting data to the device;

providing a two-wire Ethernet connection for transmitting data to and from the device;

providing a switched connection between the four-terminal Ethernet connection and the two-wire Ethernet connection, the switched connection operating in a first mode in which

the two-wire Ethernet connection is connected to the first pair of terminals and a second mode in which the two-wire Ethernet connection is connected to the second pair of terminals;

setting the switched connection to operate in the first mode;

monitoring the two-wire Ethernet connection for data being transmitted to the device; and

setting the switched connection to operate in the second mode upon detection of data being transmitted to the device.

19. The method of claim 18, further comprising injecting power on to the two-wire Ethernet connection at the device for use in powering another device connected to the two-wire Ethernet connection.

20. The method of claim 18, further comprising changing an impedance presented to the two-wire Ethernet connection at the device from a first impedance when the switched connection is operating in the first mode to a second impedance when the switched connection is operating in the second mode.

21. A field device for use with a powered Ethernet connection, the field device comprising:

communications circuitry operable to communicate data over an Ethernet connection using an Ethernet protocol;

power circuitry operable to extract operating power from the Ethernet connection; and

process circuitry operable to:

sense a process condition and to transmit information about the sensed process condition using the Ethernet connection; or

control a process condition in response to a command received through the Ethernet connection.

Appendix

22. The field device of claim 21, wherein the communications circuitry and the power circuitry are operable to interface with an Ethernet connection including only a single pair of wires.

23. A system providing a powered Ethernet connection between two devices, the system comprising:

a first device;

a second device;

an Ethernet connection between the first device and the second device, the Ethernet connection providing data communication between the devices;

circuitry for applying electrical power on to the Ethernet connection at the first device;

circuitry for extracting electrical power from the Ethernet connection at the second device; and

circuitry for powering the second device using the extracted electrical power.

24. The system of claim 23, wherein the Ethernet connection comprises two pairs of wires, with a first pair of wires being used to transmit data from the first device to the second device and a second pair of wires being used to transmit data from the second device to the first device.

25. The system of claim 24, wherein the circuitry for applying electrical power on to the Ethernet connection at the first device comprises circuitry for applying a DC voltage across the two pairs of wires by coupling a first potential to the first pair of wires and a second potential to the second pair of wires, with the DC voltage being defined as a difference between the two potentials.

26. The system of claim 25, wherein:

each pair of wires is connected to a corresponding transformer at the first device,

each transformer includes a center-tapped primary winding, and

the circuitry for coupling potentials to the pairs of wires comprises circuitry for applying the DC voltage between the center taps of the primary windings of the two transformers.

27. The system of claim 25, wherein:

a center-tapped inductor is connected across each pair of wires at the first device, and

the circuitry for coupling potentials to the pairs of wires comprises circuitry for applying the DC voltage between the center taps of the inductors.

28. The system of claim 25, wherein:

each pair of wires is connected to a corresponding transformer at the second device,

each transformer includes a center-tapped winding, and

the circuitry for extracting electrical power from the Ethernet connection at the second device comprises circuitry for extracting a DC voltage existing between the center taps of the windings of the two transformers.

29. The system of claim 25, wherein:

a center-tapped inductor is connected across each pair of wires at the second device, and

the circuitry for extracting electrical power from the Ethernet connection at the second device comprises circuitry for extracting a DC voltage existing between the center taps of the inductors.

30. The system of claim 23, wherein the Ethernet connection comprises a single pair of wires used to transmit data from the first device to the second device, to transmit data from the second device to the first device, and to provide power from the first device to the second device.

31. The system of claim 30, wherein the circuitry for applying electrical power on to the Ethernet connection at the first device comprises circuitry for applying a DC voltage across the pair of wires by coupling a first potential to the first wire through an inductor and

coupling a second potential to the second wire through an inductor, with the DC voltage being defined as a difference between the two potentials.

32. The system of claim 23, wherein the first device comprises a process controller.

33. The system of claim 32, wherein the second device comprises a field device operable to sense a process condition and to transmit information about the sensed process condition to the first device using the Ethernet connection.

34. The system of claim 32, wherein the second device comprises a field device operable to control a process condition in response to a command sent from the first device using the Ethernet connection.

35. The system of claim 23, wherein the first device comprises an Ethernet hub.

36. The system of claim 35, wherein the second device comprises a field device operable to sense a process condition and to transmit information about the sensed process condition to the first device using the Ethernet connection.

37. The system of claim 35, wherein the second device comprises a field device operable to control a process condition in response to a command sent from the first device using the Ethernet connection.

38. The system of claim 23, wherein the second device comprises a field device operable to sense a process condition and to transmit information about the sensed process condition to the first device using the Ethernet connection.

39. The system of claim 23, wherein the second device comprises a field device operable to control a process condition in response to a command sent from the first device using the Ethernet connection.

Appendix (from PCT Patent Application US00/10013, filed April 14, 2000)

ABSTRACT

A powered Ethernet connection is provided between two devices connected by an Ethernet connection that provides data communication between the devices. Electrical power is applied to the Ethernet connection at the first device, and extracted from the Ethernet connection at the second device. The extracted power is used to power the second device.

Appendix

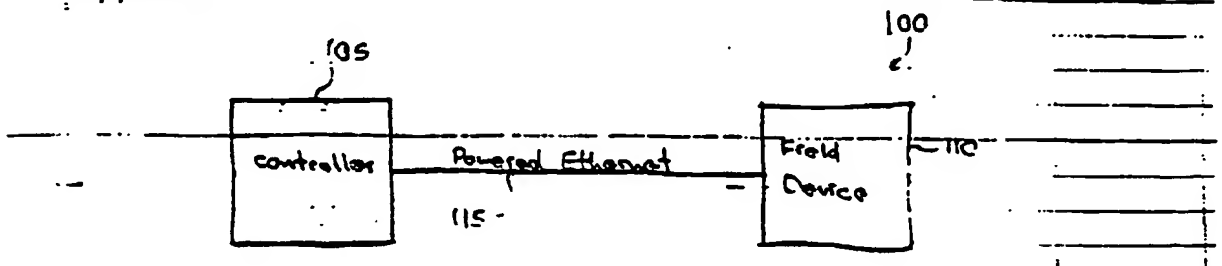


Fig. 1

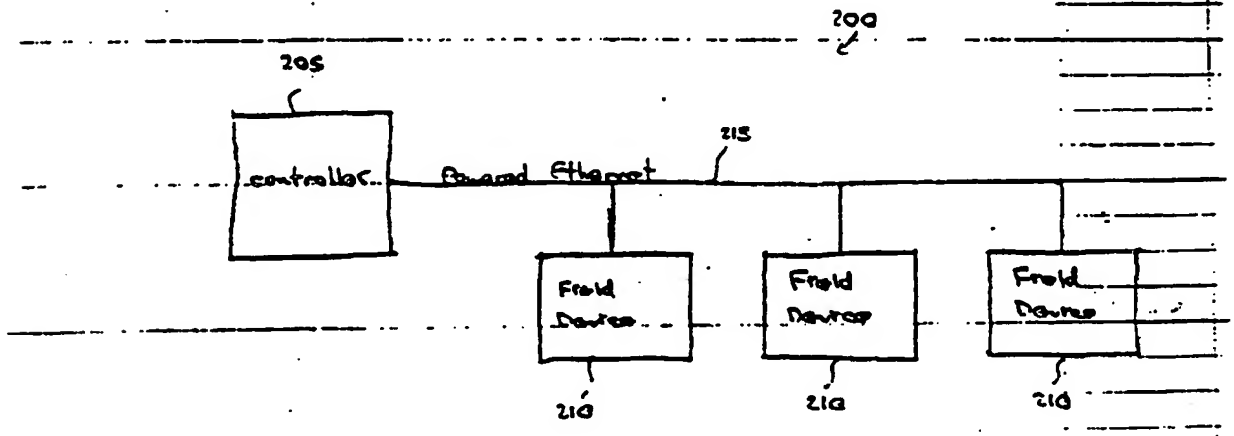


Fig. 2

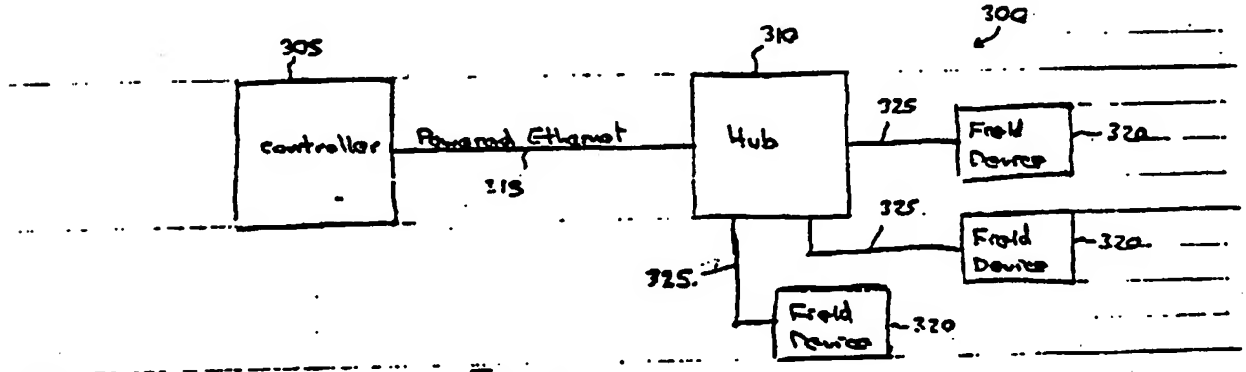


Fig. 3

Appendix

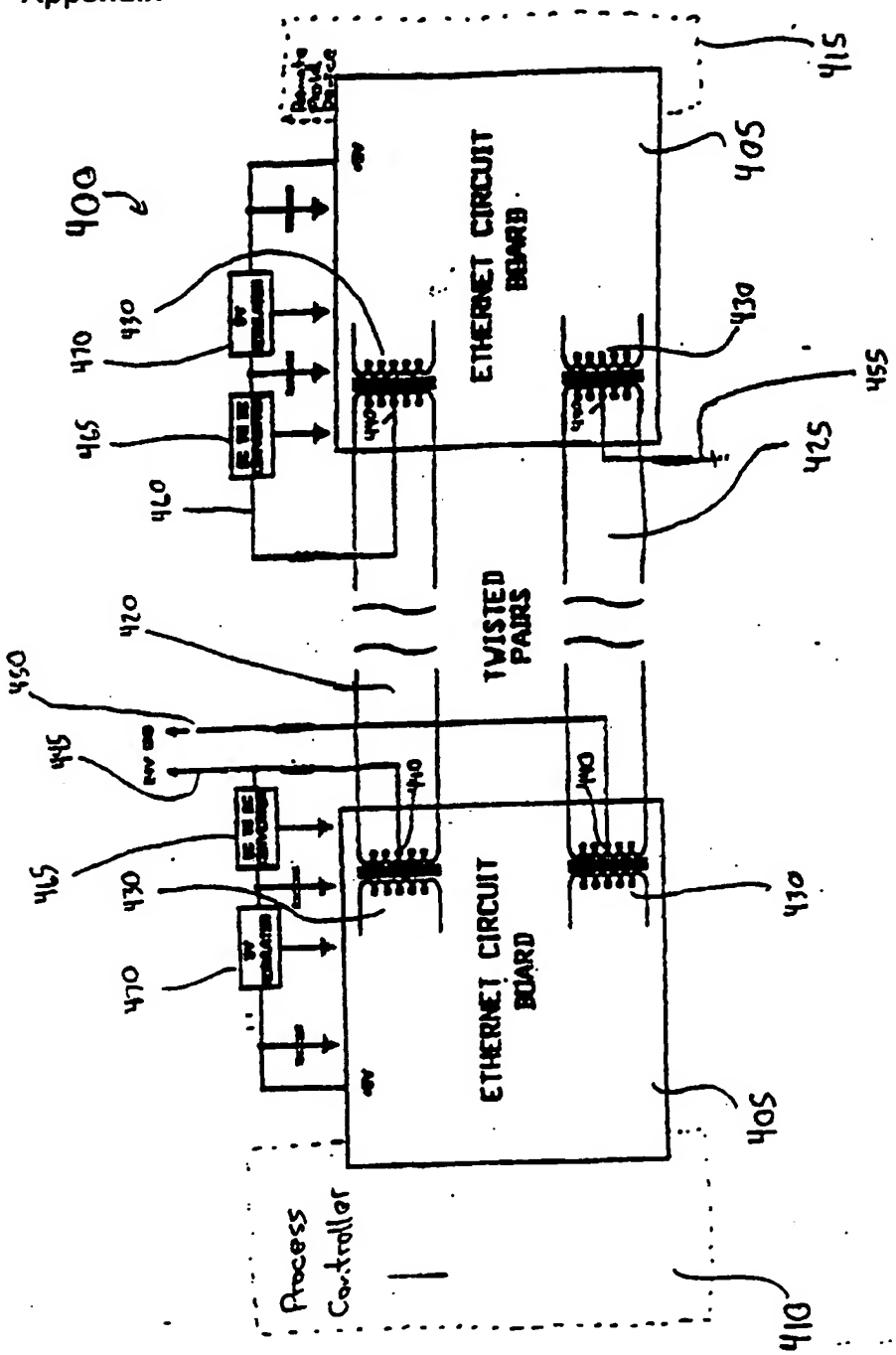


Fig. 4

Appendix

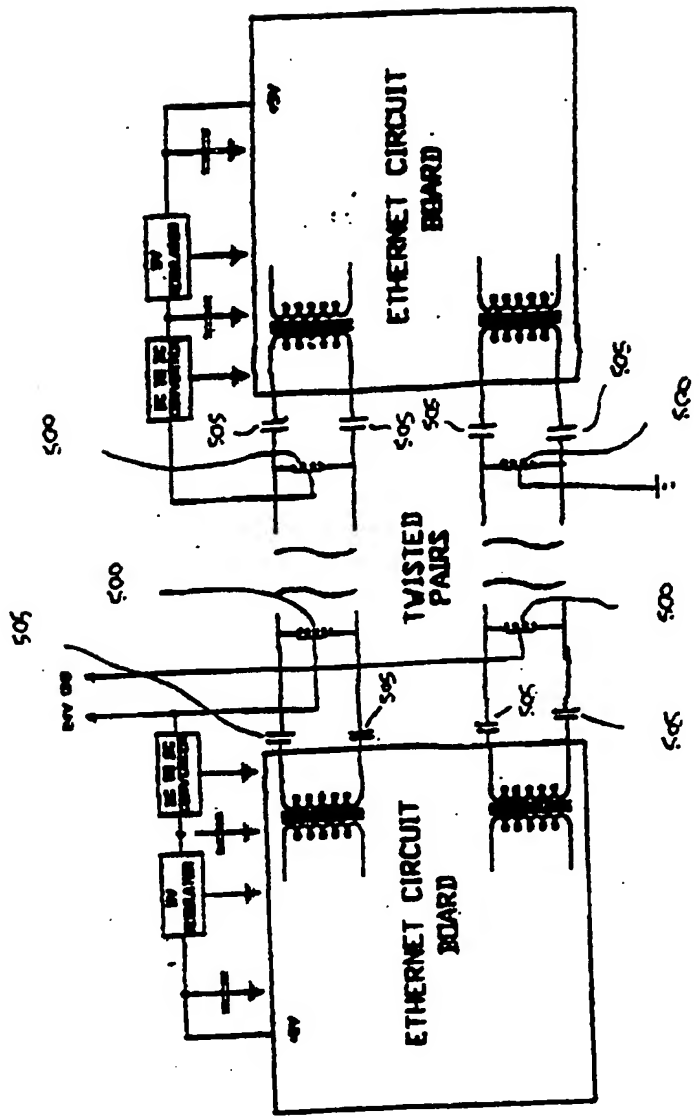
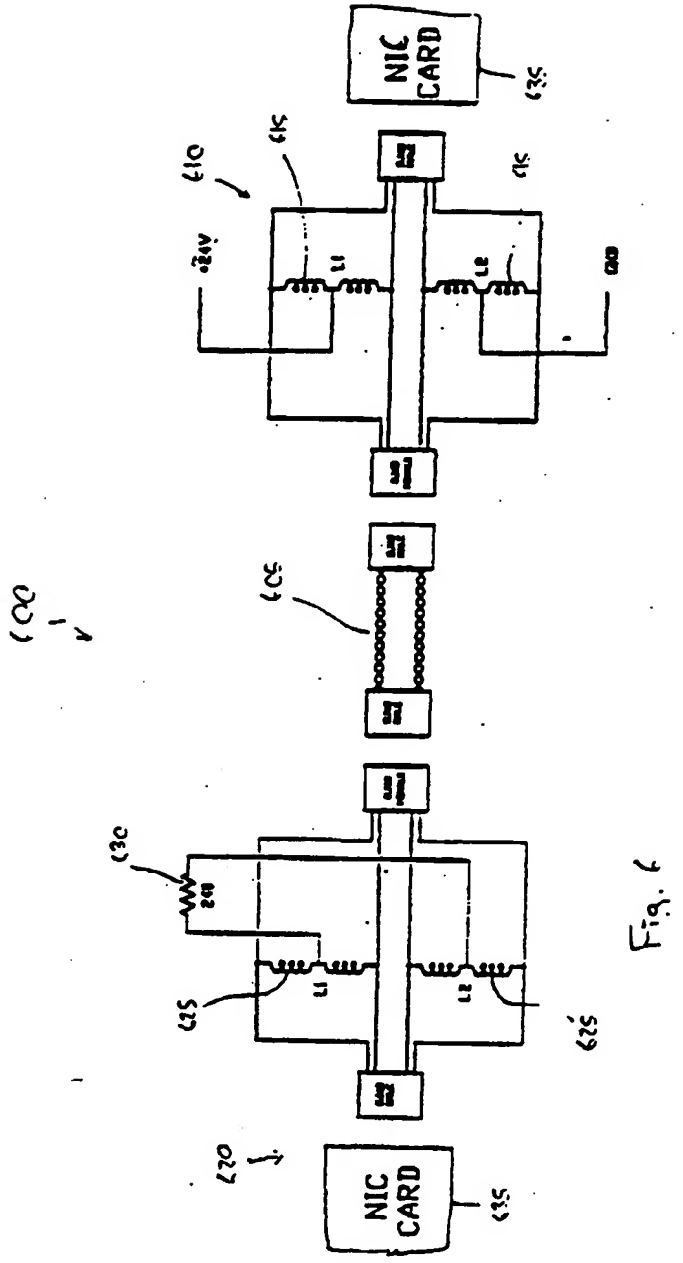


Fig. 5

Appendix


NIC CARD
NIC CARD
100
610
615
615
620
625
625
630
635
635
640
645
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840

Appendix

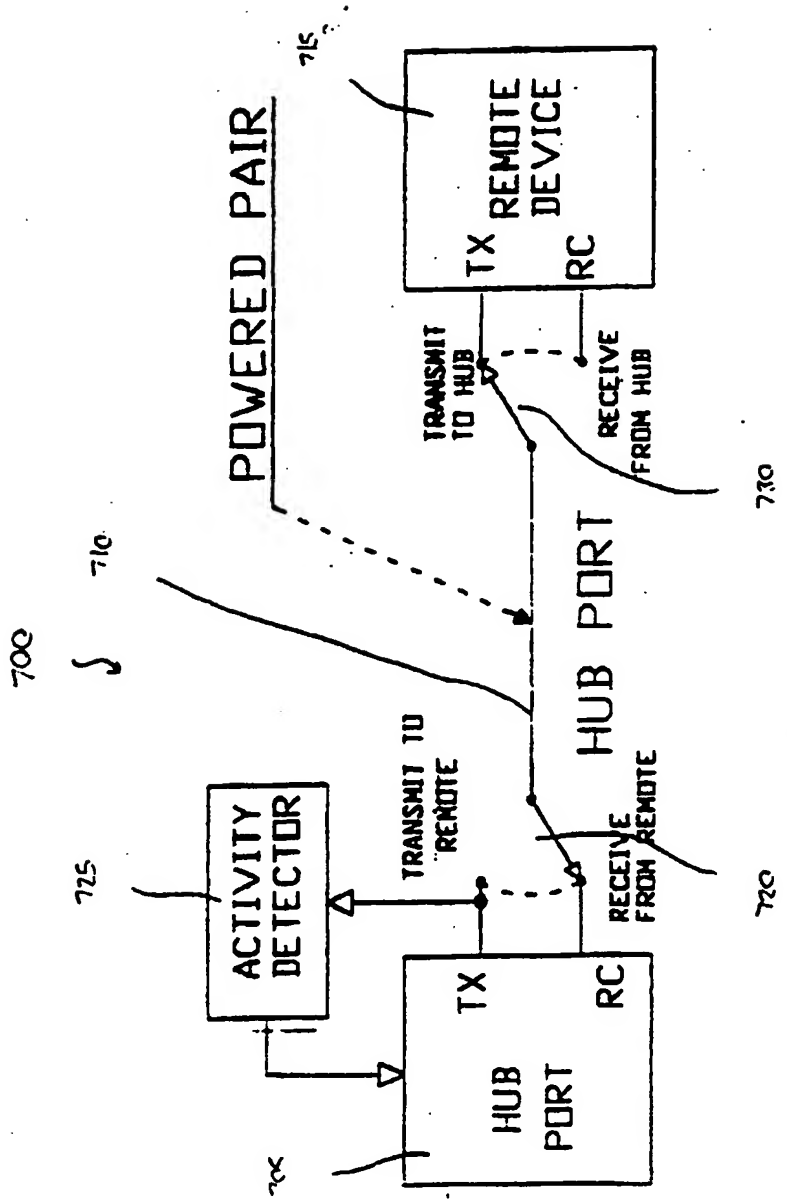


Fig. 7

Appendix

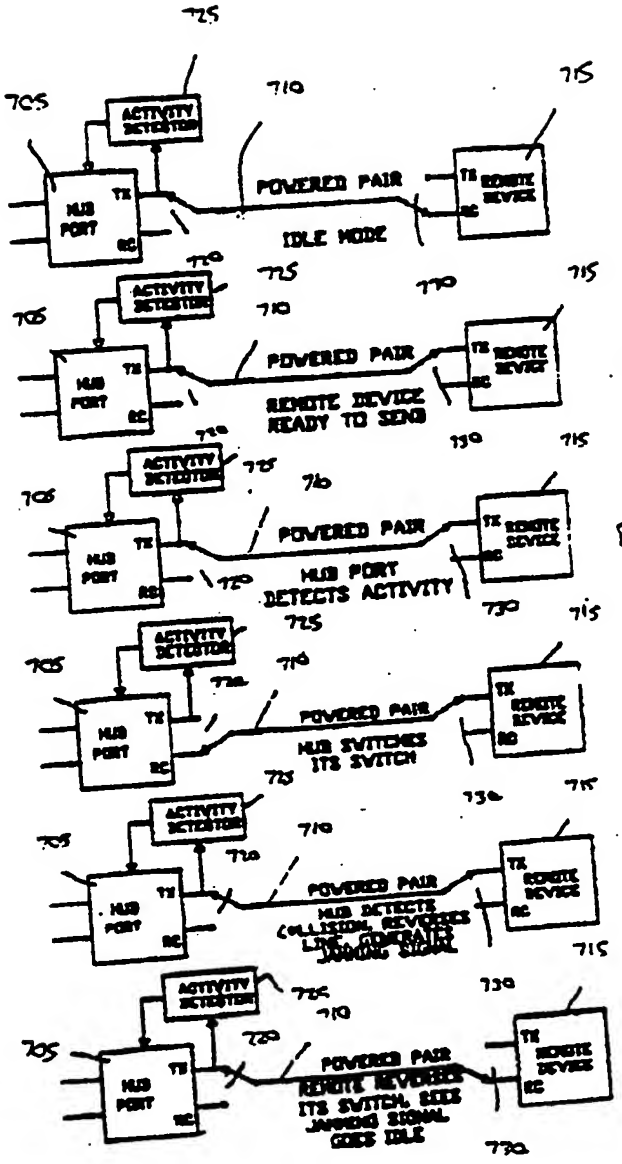


Fig. 8A

Fig. 8B

Fig. 8C

Fig. 8D

Fig. 8E

Fig. 8F

Appendix

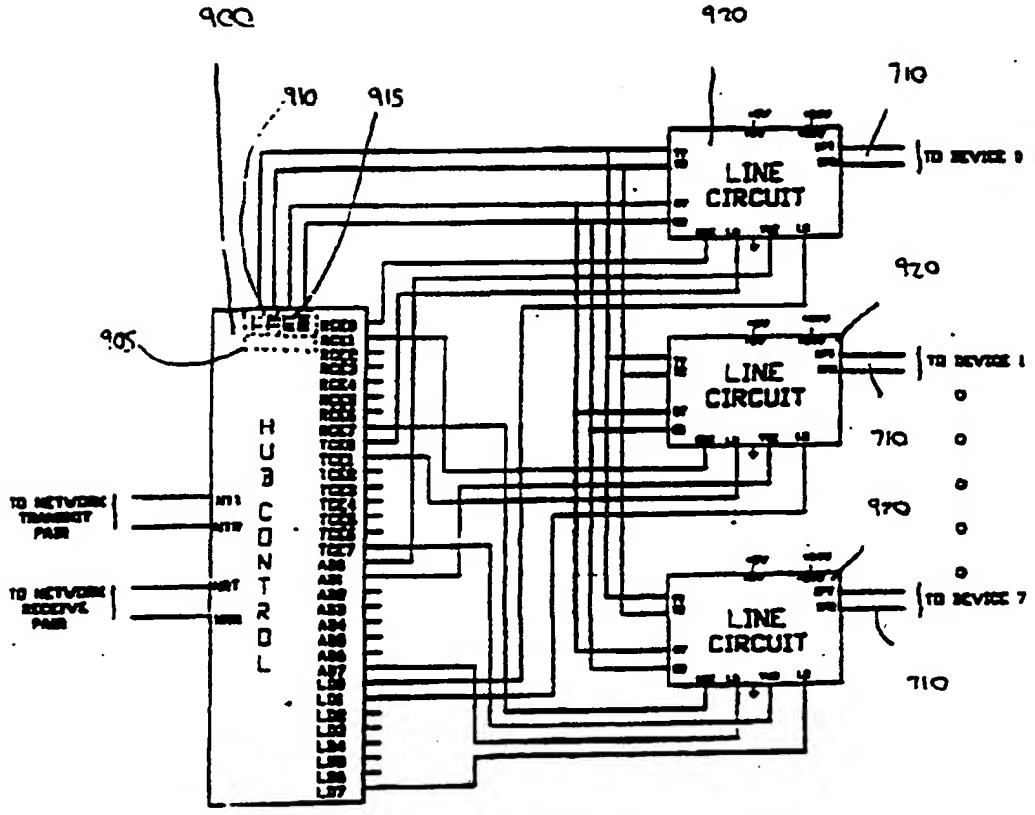


Fig. 9

Appendix

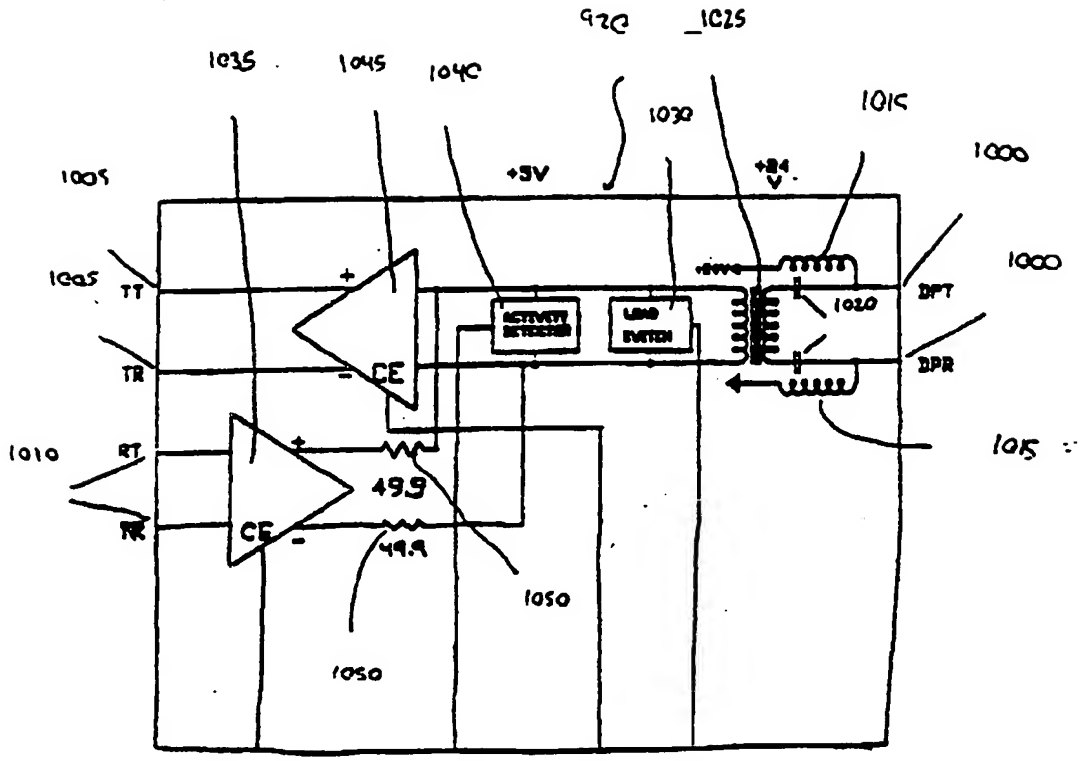


Fig. 10

Appendix

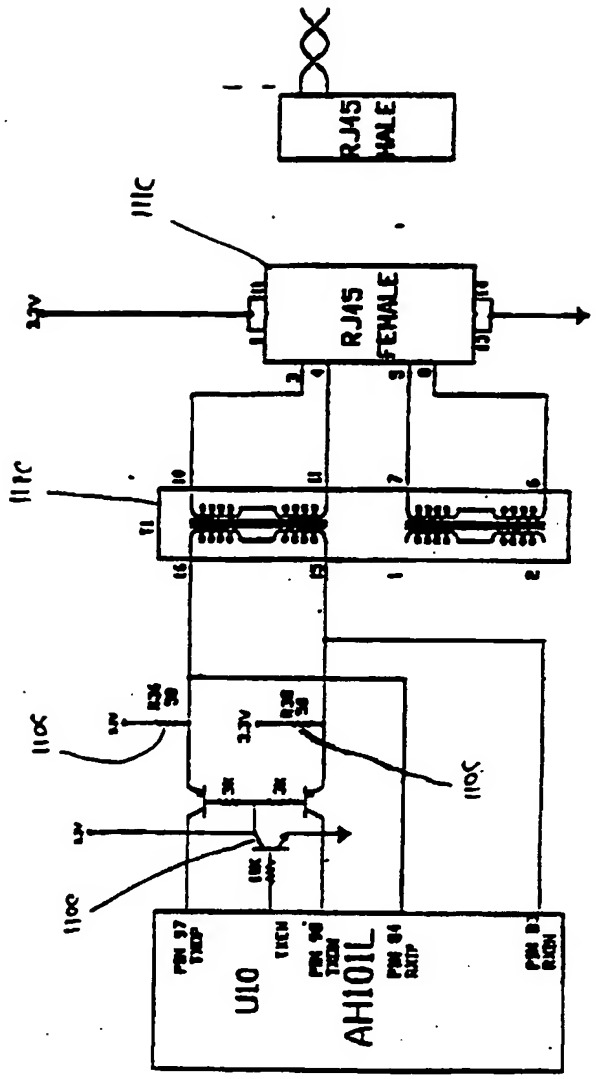


Fig. 11

Appendix

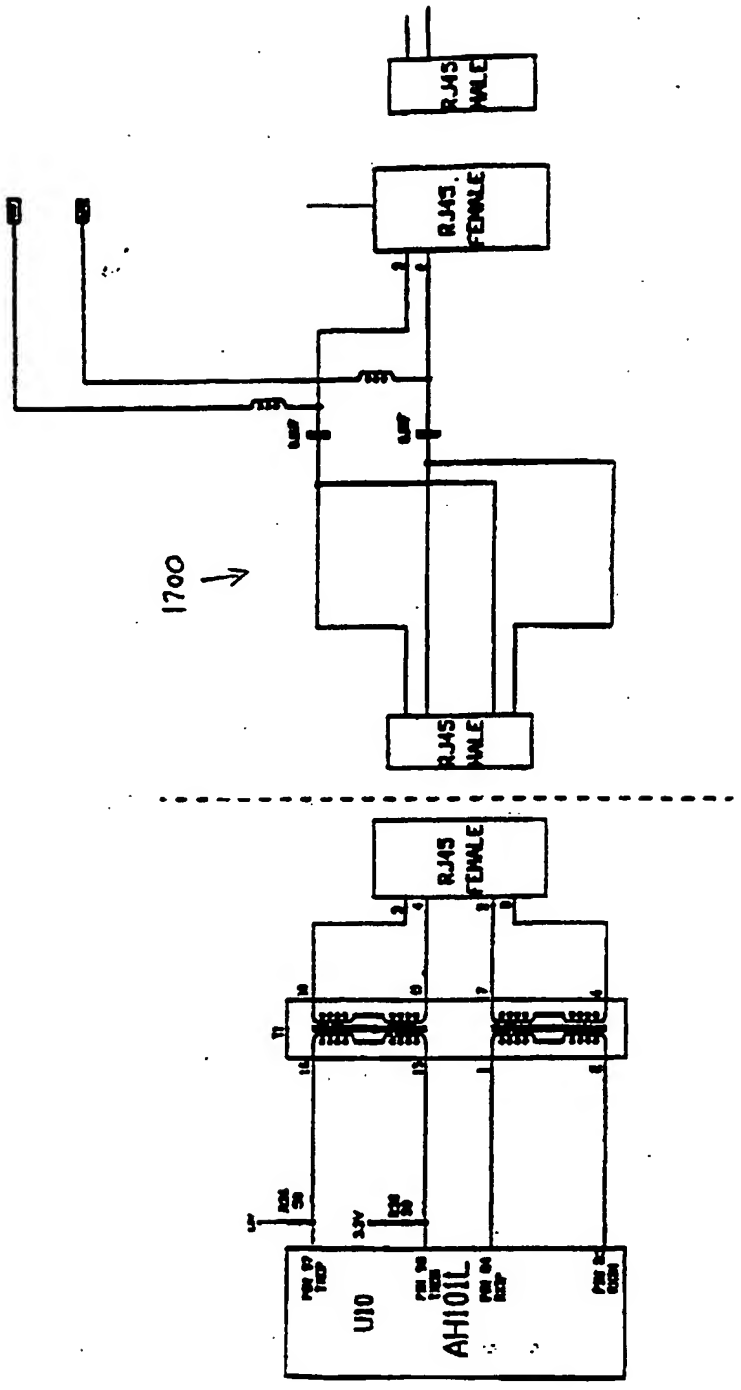


Fig. 12

Appendix

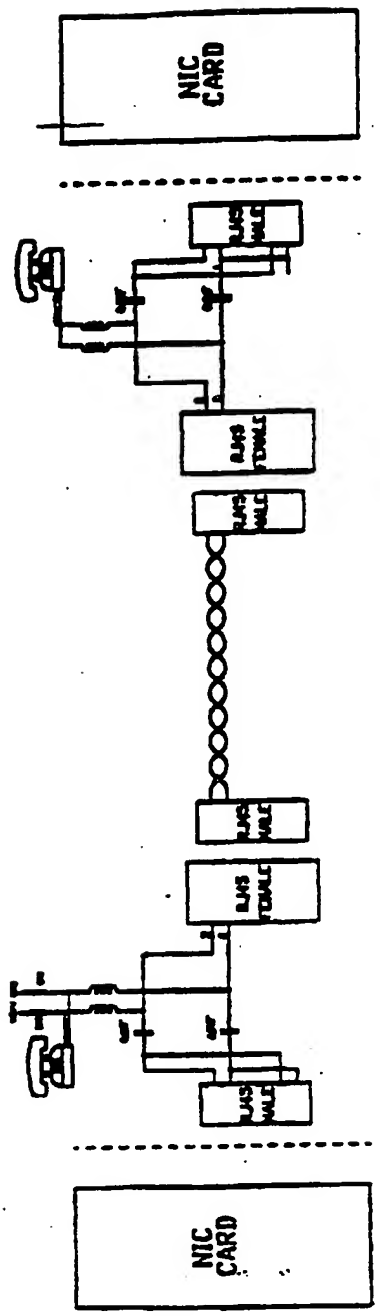
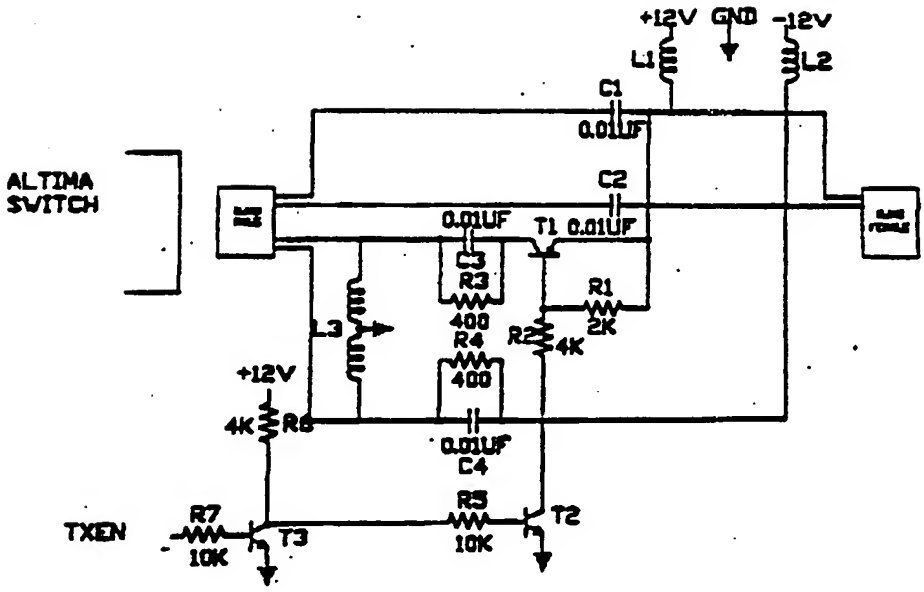


Fig. 12

Appendix

Fig. 14



Appendix

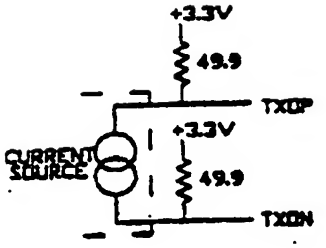


Fig. 16

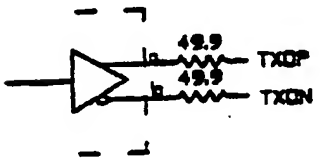
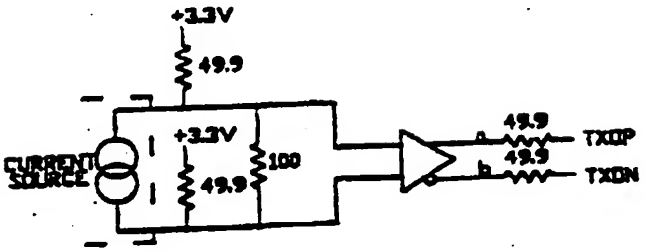


Fig. 17



Claims

1. A field device for a control system, the improvement wherein

the field device provides a virtual machine environment and executes byte code therein,

the byte code configuring the field device to execute a control algorithm.

2. The field device of claim 1, the further improvement wherein the byte code configures the field device to execute a control function block.

3. The field device of claim 1, the further improvement wherein the byte code configures the field device as a controller.

4. The field device of claim 3, the further improvement wherein the byte code configures the field device to perform proportional integral derivative control.

5. A field device for a control system, the improvement comprising

a virtual machine environment within the field device,

byte code executing within the field device that configures it to perform signal conditioning.

6. The field device of any of claims 1 - 5, the further improvement wherein the field device comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

7. The field device of any of claims 1 - 5, the further improvement wherein the byte code comprises JAVA byte code.

8. A field device for process control, the improvement wherein the field device comprises a web server.

9. The field device of claim 8, the further improvement wherein the web server is embedded.

10. The field device of claim 8, the further improvement wherein the web server serves web pages that facilitate any of configuration, monitoring and maintenance of at least the field device.

11. The field device of claim 10, the further improvement comprising a configuration editor in communication with the web server, the configuration editor controlling at least the field devices' configuration.

12. The field device of claim 11, the further improvement wherein the configuration editor is selectively disabled or enabled, depending upon the type of network to which the field device is coupled.

13. A field device for a control system, the improvement wherein the field device provides a web server and a virtual machine environment.

14. The field device of claim 13, the further improvement wherein the virtual machine environment executes byte code that configures the field device to any of (i) execute a control algorithm and (ii) perform signal conditioning.

15. The field device of claim 14, the further improvement wherein the byte code configures the field device as a controller.

16. The field device of claim 15, the further improvement wherein the byte code configures the field device to perform proportional integral derivative control.

17. The field device of claim 14, the further improvement wherein the web server is embedded.

18. The field device of claim 14, the further improvement wherein the web server facilitates any of configuration, monitoring and maintenance of at least the field device.

19. The field device of claim 18, the further improvement comprising a configuration editor that is in communication with the web server, the configuration editor controlling at least the field devices' configuration.

20. The field device of claim 19, the further improvement wherein the configuration editor is selectively disabled or enabled, depending upon the type of network to which the field device is coupled.

21. The field device of any of claims 13 - 20, the further improvement wherein the field device comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

22. The field device of any of claims 13 - 20, the further improvement wherein the byte code comprises JAVA byte code.

23. A field device for a control system, the improvement wherein the field device comprises an interface to an IP network.

24. The field device of claim 23, the further improvement wherein the interface receives operational power for the field device from the IP network.

25. The field device of claim 23, the further improvement wherein the interface comprises an interface to an Ethernet network.

26. The field device of claim 25, the further improvement wherein the interface receives operational power for the field device from the Ethernet network.

27. The field device of claim 23, the further improvement comprising a processor that is in communication with the interface.

28. The field device of claim 27, the further improvement wherein the processor is a low-power processor.

29. The field device of claim 27, the further improvement wherein the processor provides a virtual machine environment and executes a web server.

30. The field device of claim 29, the further improvement wherein the processor executes an operating system.

31. The field device of claim 30, the further improvement wherein the operating system is a real-time operating system.

32. The field device of claim 30, the further improvement wherein the field device comprises at least one of a random access memory, a read-only memory, FlashRAM, a FlashRAM, and a sensor interface.

33. The field device of any of claims 23 - 32, the further improvement wherein the field device comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

34. A field device for process control, the improvement comprising

a processor, and

an interface to an IP network, the interface being in communication with the processor.

35. The field device of claim 34, the further improvement wherein interface receives operational power for the field device from the IP network.

36. The field device of claim 34, the further improvement wherein the interface comprises an interface to an Ethernet network.

37. The field device of claim 36, the further improvement wherein the interface receives operational power from the Ethernet network.

38. The field device of claim 34, the further improvement wherein the processor executes a web server.

39. The field device of claim 34, the further improvement wherein the processor executes any of (i) a control algorithm and (ii) a signal conditioning algorithm.

40. The field device of claim 39, the further improvement wherein the processor performs proportional integral derivative control.

41. The field device of claim 34, the further improvement wherein the processor executes a web server that facilitates any of configuration, monitoring and maintenance of at least the field device.

42. The field device of claim 41, the further improvement comprising a configuration editor in communication with the web server.

43. The field device of claim 42, the further improvement wherein the configuration editor is selectively disabled or enabled, depending upon the type of network to which the field device is coupled.

44. The field device of any of claims 34 - 43, the further improvement wherein the field device comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

45. The field device of claim 44, the further improvement wherein the field device is adapted for use in a process control system.

46. A field device for a control system, the improvement wherein the field device includes an interface to an IP network and wherein the field device issues to the IP network, via the interface, a request for assignment of an IP address.

47. The field device of claim 46, the further improvement wherein the field device receives a device identification name from any of (i) a user-configured hub or other device to which the field device is coupled, (ii) a letterbug installed in the field device, (iii) a digital data processing apparatus, (iv) a software generated letterbug.

48. The field device of claim 46, the further improvement wherein the field device registers a characteristic via the IP network.

49. The field device of claim 48, the further improvement wherein the field device registers the characteristic with a bulletin board on the IP network.

50. The field device of claim 49, the further improvement wherein the field device registers the characteristic with in a Javaspace on the IP network.

51. The field device of claim 46, the further improvement wherein the field device communicates with another element of the system over the IP network in order to obtain configuration information.

52. The field device of claim 46, the further improvement wherein the field device retains configuration for use at startup.

53. The field device of any of claims 46 - 52, the further improvement wherein the field device has a processor that performs any of (i) a control algorithm and (ii) signal conditioning.

54. The field device of claim 53, the further improvement wherein the processor configures the field device as a controller.

55. The field device of claim 54, the further improvement wherein the processor configures the field device to perform proportional integral derivative control.

56. The field device of any of claims 46 - 52, the further improvement wherein the field device includes a web server that facilitates any of configuration, monitoring and maintenance of at least the field device.

57. The field device of claim 56, the further improvement wherein the field device comprises a configuration editor.

58. The field device of claim 57, the further improvement wherein the configuration editor is selectively disabled or enabled, depending upon the type of network to which the field device is coupled.

59. The field device of any of claims 46 - 52, the further improvement wherein the field device comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

60. The field device of claim 59, the further improvement wherein the field device is adapted for use in a process control system.

61. A control device for a control system, the improvement wherein

the control device provides a virtual machine environment and executes byte code therein,

the byte code configuring the control device to execute a control algorithm.

62. The control device of claim 61, the further improvement wherein the byte code configures the control device to execute a control function block.

63. The control device of claim 61, the further improvement wherein the byte code configures the control device as a controller.

64. The control device of claim 63, the further improvement wherein the byte code configures the control device to perform proportional integral derivative control.

65. The control device of any of claims 61 - 64, the further improvement wherein the control device comprises any of web server, control station, operator console, personal computer, handheld computer, workstation, integrator, controller, transmitter or other sensor device, positioner or other actuator device.

66. The control device of any of claims 61 - 64, the further improvement wherein the byte code comprises JAVA byte code.

67. A control device for a control system, the improvement wherein the control device provides a web server and a virtual machine environment.

68. The control device of claim 67, the further improvement wherein the virtual machine environment executes byte code that configures the control device to execute a control algorithm.

69. The control device of claim 68, the further improvement wherein the byte code configures the control device as a controller.

70. The control device of claim 69, the further improvement wherein the byte code configures the control device to perform proportional integral derivative control.

71. The control device of claim 67, the further improvement wherein any of the web server and the virtual machine environment is embedded.

72. The control device of claim 67, the further improvement wherein the web server any of (i) facilitates any of configuration, monitoring and maintenance of the control system or one or more control devices, (ii) collects process data from one or more control devices, (iii) generates source for operator displays, (iv) provides access to the control system, and (v) hosts an applications development environment.

73. The control device of any of claims 67 - 72, the further improvement wherein the control device comprises any of a web server, control station, operator console, personal computer, handheld computer, workstation, integrator, controller, transmitter or other sensor device, positioner or other actuator device.

74. The control device of any of claims 67 - 72, the further improvement wherein the byte code comprises JAVA byte code.

75. A control device for a control system, the improvement wherein the control device comprises

a low-power processor,

an interface to an IP network, the interface being in communication with the processor, and

the interface receives operational power for the control device from the IP network.

76. The control device of claim 75, the further improvement wherein the interface comprises an interface to an Ethernet network.

77. The control device of claim 75, the further improvement wherein the processor provides a virtual machine environment and executes a web server.

78. The control device of claim 77, the further improvement wherein the control device comprises at least one of a random access memory, a read-only memory, FlashRAM, and a sensor interface, access to permanent storage, a configurator, system management software, messaging services, alarm/event notification, byte code implementing process control functions, byte code implementing status reporting.

79. The control device of any of claims 75 - 78, the further improvement wherein the control device comprises any of a web server, control station, operator console, personal computer, handheld computer, workstation, integrator, controller, transmitter or other sensor device, positioner or other actuator device.

80. A control device for process control, the improvement wherein

the device comprises an interface to an IP network, and

on startup, the device registers a characteristic thereof with at least one other device on the IP network.

81. The control device of claim 80, the further improvement wherein the control device issues to the IP network, via the interface, a request for assignment of an IP address.

82. The control device of claim 80, the further improvement wherein the control device receives a device identification name from any of (i) a user-configured hub or other device to which the control device is coupled, (ii) a letterbug installed in the control device, (iii) a digital data processing apparatus, (iv) a software generated letterbug.

83. The control device of claim 80, the further improvement wherein the control device registers the characteristic with a bulletin board on the IP network.

84. The control device of claim 83, the further improvement wherein the control device registers the characteristic with in a Javaspace on the IP network.

85. The control device of claim 80, the further improvement wherein the control device communicates with another device over the IP network in order to obtain configuration information.

86. The control device of claim 80, the further improvement wherein the control device retains configuration information for use at startup.

87. The control device of any of claims 81 - 86, the further improvement wherein the control device has a processor that executes code that performs a control algorithm.

88. The control device of claim 87, wherein the processor executes code that configures the control device as a controller.

89. The control device of claim 88, the further improvement wherein the processor executes code to perform proportional integral derivative control.

90. The control device of any of claims 81 - 86, the further improvement wherein the control device includes a web server that any of (i) facilitates any of configuration, monitoring and maintenance of the control system or one or more control devices, (ii) collects process data from one or more control devices, (iii) generates source for operator displays, (iv) provides access to the control system, and (v) hosts an applications development environment.

91. The control device of claim 90, the further improvement wherein the control device comprises a configuration editor.

92. The control device of any of claims 81 - 86, the further improvement wherein the control device comprises any of a web server, control station, operator console, personal computer, handheld computer, workstation, integrator, controller, transmitter or other sensor device, positioner or other actuator device.

93. The control device of claim 92, the further improvement wherein the control device is adapted for use in a process control system.

94. A control system comprising one or more field devices or control devices according to any of claims 1 - 5, 8 - 20, 23 - 32, 34 - 43, 46 - 53, 61 - 64, 67 - 72, 75 - 78, 80 - 86, and 88 coupled via one or more IP networks.

95. A control system according to claim 94, wherein the IP network is powered.

96. A process control system having one or more field devices according to any of claims 8 - 12.

97. A control system comprising

a plurality of control devices coupled for communication via an IP network,

a DHCP server that furnishes IP addresses in response to requests by one or more of the control devices.

98. The control system of claim 97, wherein one or more of the control devices are field devices.

99. The control system of claim 98, adapted for process control, wherein the control devices are process control devices.

100. The control system of any of claims 97 - 99, wherein the control devices comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

101. The control system of any of claims 97 - 99, wherein the DHCP server is solid state.

102. The process control system of claim 101, wherein the DHCP server is free of moving parts and comprises zero, one or more removable components.

103. The control system of any of claims 97 - 99, wherein the IP network is powered and wherein one or more of the control devices receive operational power from the IP network.

104. A DHCP server that is solid state and that is adapted for use on an IP network comprising one or more control devices.

105. The solid state DHCP server of claim 104, adapted for use with one or more control devices that are process control devices.

106. The solid state DHCP server of claim 104, adapted for use with one or more control devices that are field devices.

107. The solid state DHCP server of claim 106, adapted for use with field devices that include any of a transmitter or other sensor device, and a positioner or other actuator device.

108. A control system comprising

a plurality of control devices coupled for communication via an IP network,

a network enabler that is coupled to the IP network that responds to requests by the control devices to at least one of

i) register names specified by the control devices,

ii) search for names specified by the control devices,

iii) posting to a network bulletin board events specified by the control devices,

v) removing from the network bulletin board events specified by the control devices,

vi) querying the network bulletin board for events specified by the control devices,

vii) notifying the control devices of events specified by the control devices.

109. The control system of claim 108, wherein the network enabler is any of a JINI and a JavaSpace server.

110. The control system of claim 108, wherein one or more of the control devices are field devices.

111. The control system of claim 110, adapted for process control, wherein the control devices are process control devices.

112. The control system of any of claims 108 - 111, wherein the control devices comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

113. The control system of any of claims 108 - 111, wherein the network enabler is solid state.

114. The process control system of claim 113, wherein the network enabler is free of moving parts and comprises zero, one or more removable components.

115. The control system of any of claims 108 - 111. wherein the IP network is powered and wherein one or more of the control devices receive operational power from the IP network.

116. A network enabler comprising

at least one IP network interface, and

one or more server functionalities, each of which responds to requests received over the network interface to at least one of

i) register a name specified in a request,

ii) search for a name specified in a request,

iii) post to a network bulletin board an event specified in a request,

v) remove from the network bulletin board an event specified in a request,

vi) query the network bulletin board for an event specified in a request,

vii) notify a requestor of events specified thereby,

viii)serve as a web server.

117. The network enabler of claim 116, wherein the server functionality is any of a JINI and a JavaSpace server.

118. The network enabler of any of claims 116 - 117, wherein the network enabler is solid state.

119. The network enabler of claim 118, wherein the network enabler is free of moving parts and comprises zero, one or more removable components.

120. The network enabler of claim 118, wherein the network enabler receives operational power from the IP network.

121. A method of operating a field device for a control system, the improvement comprising the steps of

providing within the field device a virtual machine environment,

executing byte code embodying a control algorithm within the virtual machine environment.

122. The method of claim 121, the further improvement comprising executing a control function block with the byte code.

123. The method of claim 121, the further improvement comprising configuring the field device as a controller.

124. The method of claim 123, the further improvement comprising configuring the field device with the byte code to perform proportional integral derivative control.

125. A method of operating a field device for a control system, the improvement comprising the steps of the steps of

providing a virtual machine environment within the field device ,

executing byte code within the virtual machine environment to configure the field device to perform signal conditioning.

126. The method of any of claims 121 - 125, the further improvement wherein the field device comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

127. The method of any of claims 121 - 125, the further improvement wherein the byte code comprises JAVA byte code.

128. A method of operating a field device for process control, the improvement comprising executing a web server within the field device.

129. The method of claim 128, the further improvement wherein the web server is embedded.

130. The method of claim 128, the further improvement comprising serving with the web server web pages that facilitate any of configuration, monitoring and maintenance of at least the field device.

131. The method of claim 130, the further improvement comprising controlling configuration of at least the field device with a configuration editor that utilizes the web server as an interface.

132. The method of claim 131, the further improvement comprising selectively disabling or enabling the configuration editor, depending upon the type of network to which the field device is coupled.

133. A method of operating a field device for a control system, the improvement comprising the steps of executing a web server and a virtual machine environment within the field device.

134. The method of claim 133, the further improvement comprising executing byte code within the virtual machine environment that configures the field device to any of (i) execute a control algorithm and (ii) perform signal conditioning.

135. The method of claim 134, the further improvement comprising configuring the field device as a controller.

136. The method of claim 135, the further improvement comprising configuring the field device with the byte code to perform proportional integral derivative control.

137. The method of claim 134, the further improvement wherein the web server is embedded.

138. The method of claim 134, the further improvement comprising facilitating any of configuration, monitoring and maintenance of at least the field device with the web server.

139. The method of claim 138, the further improvement comprising controlling at least the field devices' configuration with a configuration editor that utilizes the web server as an interface.

140. The method of claim 139, the further improvement comprising selectively disabling or enabling the configuration editor, depending upon the type of network to which the field device is coupled.

141. The method of any of claims 133 - 140, the further improvement wherein the field device comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

142. The method of any of claims 133 - 140, the further improvement wherein the byte code comprises JAVA byte code.

143. A method of operating a field device for a control system, the improvement comprising the steps of interfacing the field device with other elements of the control system via IP network.

144. The method of claim 143, the further improvement comprising receiving operational power for the field device from the IP network.

145. The method of claim 143, the further improvement comprising interfacing the field device with other elements of the control system via an Ethernet network.

146. The method of claim 145, the further improvement comprising receiving operational power for the field device from the Ethernet network.

147. The method of claim 143, the further improvement comprising providing a processor in the field device that is in communication with the IP network.

148. The method of claim 147, the further improvement wherein the processor is a low-power processor.

149. The method of claim 147, the further improvement comprising executing a virtual machine environment and a web server with the processor.

150. The method of claim 149, the further improvement comprising executing an operating system with the processor..

151. The method of claim 150, the further improvement comprising executing a real-time operating system with the processor.

152. The method of claim 150, the further improvement wherein the field device comprises at least one of a random access memory, a read-only memory, FlashRAM, and a sensor interface.

153. The method of any of claims 143 - 152, the further improvement wherein the field device comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

154. A method of operating a field device for process control, the improvement comprising the steps of

interfacing the field device with other elements of the control system via an IP network,

operating a processor in the field device that is in communication with the IP network.

155. The method of claim 154, the further improvement comprising receiving operational power for the field device from the IP network.

156. The method of claim 154, the further improvement comprising interfacing the field device with other elements of the control system via an Ethernet network.

157. The method of claim 156, the further improvement comprising receiving operational power for the field device from the Ethernet network.

158. The method of claim 154, the further improvement comprising executing a web server with the processor.

159. The method of claim 154, the further improvement comprising operating the processor to execute any of (i) a control algorithm and (ii) a signal conditioning algorithm.

160. The method of claim 159, the further improvement comprising performing proportional integral derivative control with the processor.

161. The method of claim 154, the further improvement comprising executing with the processor a web server that facilitates any of configuration, monitoring and maintenance of at least the field device.

162. The method of claim 161, the further improvement comprising controlling configuration of at least the field device with a configuration editor that utilizes the web server as an interface.

163. The method of claim 162, the further improvement comprising selectively disabling or enabling the configuration editor, depending upon the type of network to which the field device is coupled.

164. The method of any of claims 154 - 163, the further improvement wherein the field device comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

165. The method of claim 164, the further improvement comprising utilizing the field processor for process control.

166. A method of operating a field device for a control system, the improvement comprising the steps of

interfacing the field device with other elements of the control system via an IP network,

issuing a request from the field device to the IP network for an IP address.

167. The method of claim 166, the further improvement comprising receiving a device identification name from any of (i) a user-configured hub or other device to which the field device is coupled, (ii) a letterbug installed in the field device, (iii) a digital data processing apparatus, (iv) a software generated letterbug.

168. The method of claim 166, the further improvement comprising registering a characteristic of the field device via the IP network.

169. The method of claim 168, the further improvement comprising registering the characteristic with a bulletin board on the IP network.

170. The method of claim 169, the further improvement comprising registering the characteristic with in a Javaspace on the IP network.

171. The method of claim 166, the further improvement comprising communicating between the field and another element of the system over the IP network in order to obtain configuration for the field device.

172. The method of claim 166, the further improvement comprising retaining configuration information within the field device for use at startup.

173. The method of any of claims 166 - 172, the further improvement comprising performing with a processor in the field device any of (i) a control algorithm and (ii) signal conditioning.

174. The method of claim 173, the further improvement comprising configuring the field device as a controller.

175. The method of claim 174, the further improvement comprising configuring the field device to perform proportional integral derivative control.

176. The method of any of claims 166 - 172, the further improvement comprising executing a web server within the field device, the web server facilitating any of configuration, monitoring and maintenance of at least the field device.

177. The method of claim 176, the further improvement comprising executing a configuration editor within the field device and providing an interface to the configuration editor via the web server.

178. The method of claim 177, the further improvement comprising selectively disabling or enabling the configuration editor, depending upon the type of network to which the field device is coupled.

179. The method of any of claims 166 - 172, the further improvement wherein the field device comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

180. The method of claim 179, the further improvement comprising utilizing the field processor for process control.

181. A method of operating a control device for a control system, the improvement comprising the steps of

executing byte code within a virtual machine environment provided in the control device,

the byte code configuring the control device to execute a control algorithm.

182. The method of claim 181, the further improvement comprising using the byte code to configure the control device to execute a control function block.

183. The method of claim 181, the further improvement comprising using the byte code to configure the control device as a controller.

184. The method of claim 183, the further improvement comprising using the byte code to configure the control device to perform proportional integral derivative control.

185. The control device of any of claims 181 - 184, the further improvement wherein the control device comprises any of web server, control station, operator console, personal computer, handheld computer, workstation, integrator, controller, transmitter or other sensor device, positioner or other actuator device.

186. The control device of any of claims 181 - 184, the further improvement wherein the byte code comprises JAVA byte code.

187. A method of operating a control device for a control system, the improvement comprising executing a web server and a virtual machine environment within the control device.

188. The method of claim 187, the further improvement comprising using the byte code to configure the control device to execute a control algorithm.

189. The method of claim 188, the further improvement comprising using the byte code to configure the control device as a controller.

190. The method of claim 189, the further improvement comprising using the byte code to configure the control device to perform proportional integral derivative control.

191. The method of claim 187, the further improvement wherein any of the web server and the virtual machine environment is embedded.

192. The method of claim 187, the further improvement comprising using the web server to any of (i) facilitate any of configuration, monitoring and maintenance of the control system or one or more control devices, (ii) collect process data from one or more control devices, (iii) generate source for operator displays, (iv) provide access to the control system, and (v) host an applications development environment.

193. The control device of any of claims 187 - 192, the further improvement wherein the control device comprises any of a web server, control station, operator console, personal

computer, handheld computer, workstation, integrator, controller, transmitter or other sensor device, positioner or other actuator device.

194. The control device of any of claims 187 - 192, the further improvement wherein the byte code comprises JAVA byte code.

195. A method of operating a control device for a control system, the improvement comprising the steps of

operating a low-power processor within the control device,

communicating between the control device and other elements of the control system via an IP network,

drawing operational power for the control device from the IP network.

196. The method of claim 195, the further improvement comprising interfacing the field device with other elements of the control system via an Ethernet network.

197. The method of claim 195, the further improvement comprising executing a virtual machine environment and a web server with the processor.

198. The method of claim 197, the further improvement wherein the control device comprises at least one of a random access memory, a read-only memory, FlashRAM, and a sensor interface, access to permanent storage, a configurator, system management software, messaging services, alarm/event notification, byte code implementing process control functions, byte code implementing status reporting.

199. The control device of any of claims 195 - 198, the further improvement wherein the control device comprises any of a web server, control station, operator console, personal

computer, handheld computer, workstation, integrator, controller, transmitter or other sensor device, positioner or other actuator device.

200. A method of operating a control device for process control, the improvement comprising the steps of

communicating between the control device and other elements of the control system via an IP network, and

causing the control device, on startup, to register a characteristic thereof with at least one other device over the IP network.

201. The method of claim 200, the further improvement comprising issuing with the control device, via the IP network, a request for assignment of an IP address.

202. The method of claim 200, the further improvement comprising receiving for the control device a device identification name from any of (i) a user-configured hub or other device to which the control device is coupled, (ii) a letterbug installed in the control device, (iii) a digital data processing apparatus, (iv) a software generated letterbug.

203. The method of claim 200, the further improvement comprising registering a characteristic of the control device with a bulletin board on the IP network.

204. The method of claim 203, the further improvement comprising registering the characteristic with in a Javaspace on the IP network.

205. The method of claim 200, the further improvement comprising communicating between the control device and another device over the IP network in order to obtain configuration information for the control device.

206. The method of claim 200, the further improvement comprising retaining configuration information within the control device for use at startup.

207. The control device of any of claims 201 - 206, the further improvement comprising executing code embodying a control algorithm on a processor within the control device.

208. The method of claim 207, comprising using the code to configure the control device as a controller.

209. The method of claim 208, the further improvement wherein the processor executes code to perform proportional integral derivative control.

210. The control device of any of claims 201 - 206, the further improvement comprising executing a web server within the control device, the web server any of (i) facilitating any of configuration, monitoring and maintenance of the control system or one or more control devices, (ii) collecting process data from one or more control devices, (iii) generating source for operator displays, (iv) provides access to the control system, and (v) hosting an applications development environment.

211. The method of claim 210, the further improvement comprising executing a configuration editor within the control device, the configuration editor utilizing the web server as an interface.

212. The control device of any of claims 201 - 206, the further improvement wherein the control device comprises any of a web server, control station, operator console, personal computer, handheld computer, workstation, integrator, controller, transmitter or other sensor device, positioner or other actuator device.

213. The method of claim 212, the further improvement comprising utilizing the control device for process control.

214. A method of operating control system comprising the steps of

communicating among a plurality of control devices via an IP network,

generating, within at least a selected one of the control devices, a request for an IP address,

furnishing IP addresses to a requesting control device with a DHCP server in response to such a request.

215. The method of operating a control system of claim 214, wherein one or more of the control devices are field devices.

216. The method of operating a control system of claim 215, including the step of utilizing the control devices for process control.

217. The method of operating a control system of any of claims 214 - 216, wherein the control devices comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

218. The method of operating a control system of any of claims 214 - 216, comprising the step of providing a DHCP server that is solid state.

219. The process control system of claim 218, comprising providing a DHCP server that is free of moving parts and that comprises zero, one or more removable components.

220. The method of operating a control system of any of claims 214 - 216, comprising the step of drawing operational power for the control devices from the IP network.

221. A method of operating a control system comprising the steps of

communicating among a plurality of control devices coupled via an IP network,

responding to requests issued by the control devices over the IP network with a network enabler that at least one of

i) registers names specified by the control devices,

ii) searches for names specified by the control devices,

iii) posts to a network bulletin board events specified by the control devices,

v) removes from the network bulletin board events specified by the control devices,

vi) queries the network bulletin board for events specified by the control devices,

vii) notifies the control devices of events specified by the control devices.

222. The method of operating a control system of claim 221, wherein the network enabler is any of a JINI and a JavaSpace server.

223. The method of operating a control system of claim 221, wherein one or more of the control devices are field devices.

224. The method of operating a control system of claim 223, for process control.

225. The method of operating a control system of any of claims 221 - 224, wherein the control devices comprises any of a transmitter or other sensor device, and a positioner or other actuator device.

226. The method of operating a control system of any of claims 221 - 225, comprising the step of providing a network enabler that is solid state.

227. The process control system of claim 226, comprising providing a network enabler that is free of moving parts and comprises zero, one or more removable components.

228. The method of operating a control system of any of claims 221 - 224, comprising the step of drawing operational power for the control devices from the IP network.

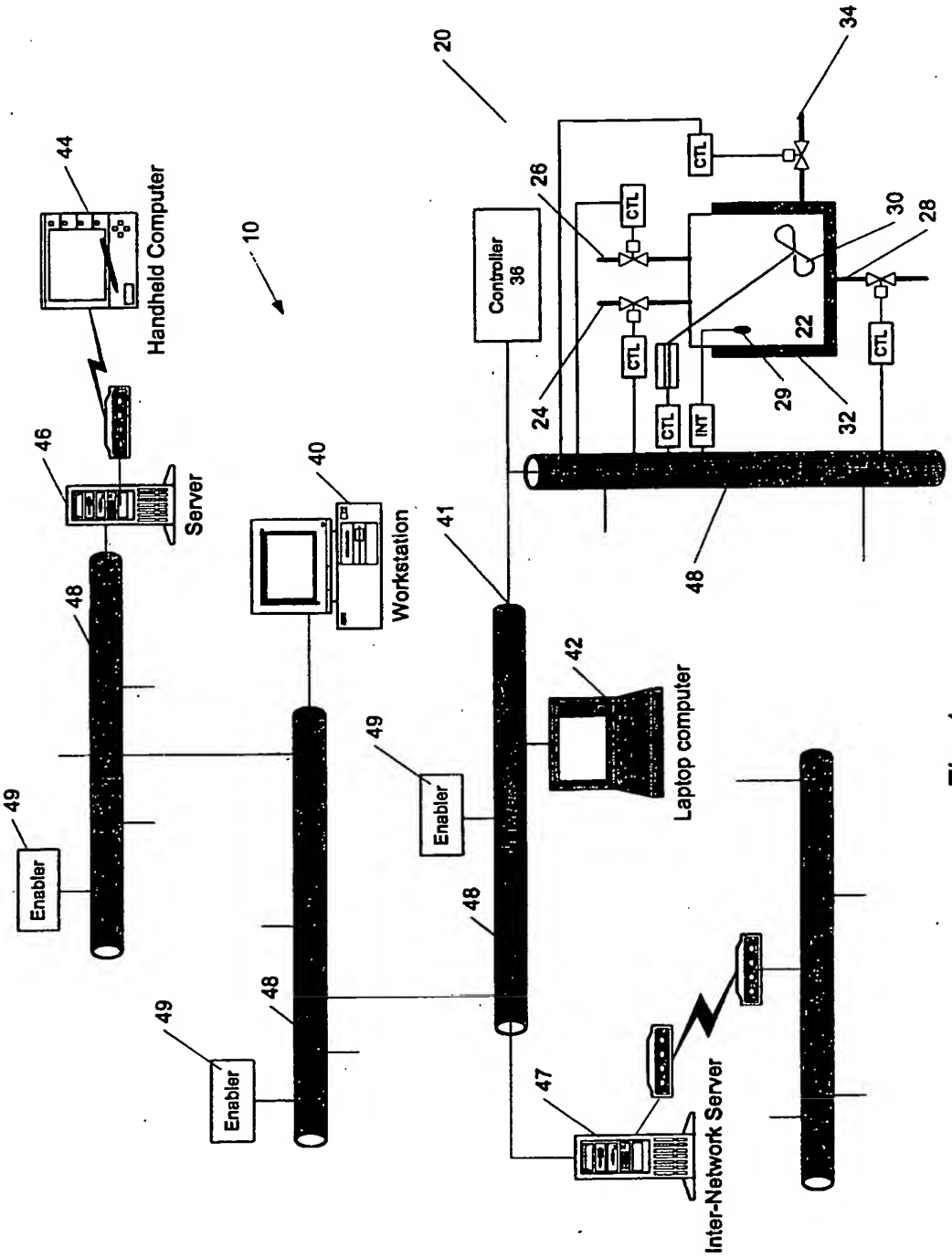

10
20
34
44
Handheld Computer
26
Controller
36
30
28
CTL
22
24
29
32
INT
46
Server
40
Workstation
41
48
42
Laptop computer
49
Enabler
47
Inter-Network Server

Fig. 1

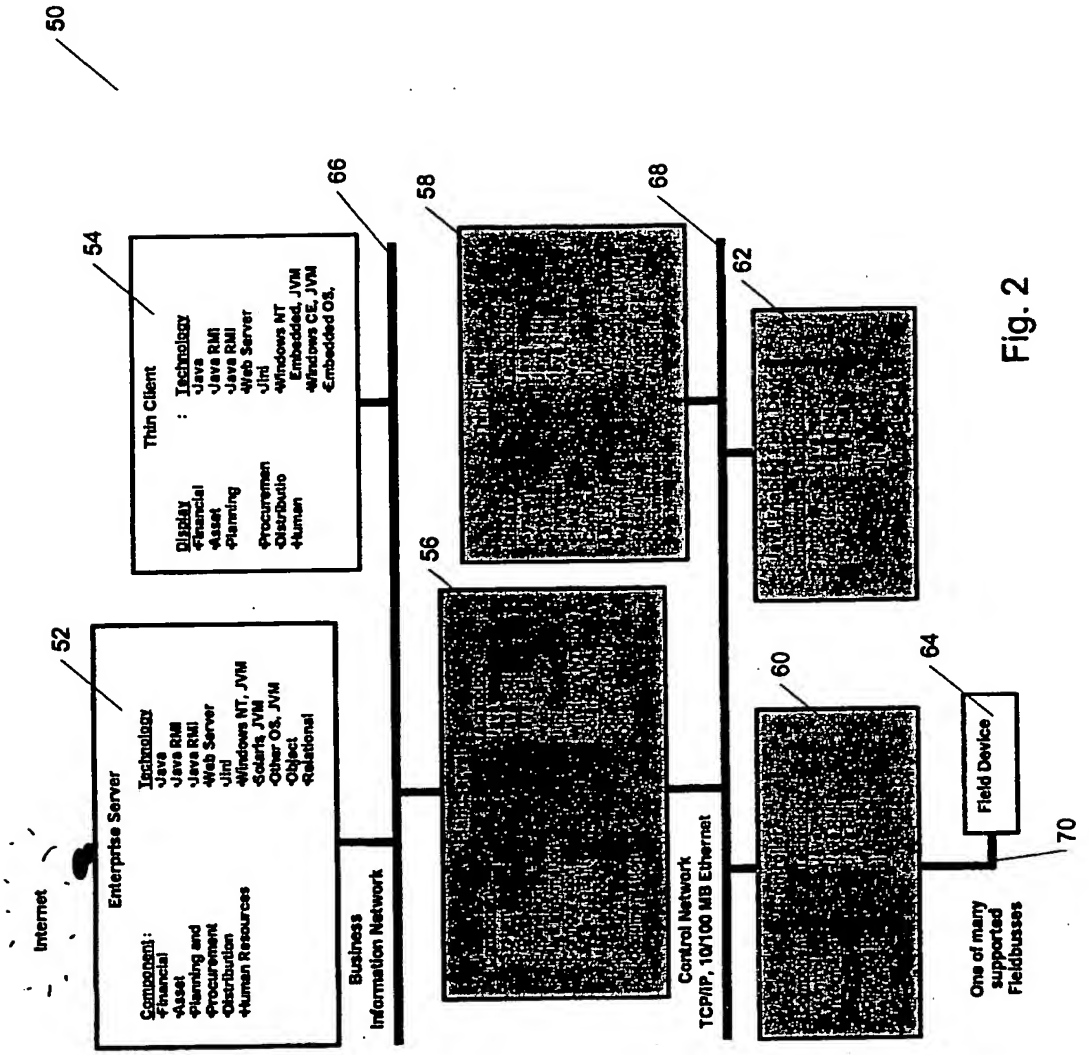

50
Internet
52
Enterprise Server
Component :
•Financial
•Asset
•Planning and
•Procurement
•Distribution
•Human Resources
Technology
•Java
•Java RMI
•Java RMI
•Web Server
•Jini
•Windows NT, JVM
•Solaris, JVM
•Other OS, JVM
•Object
•Relational
Business
Information Network
54
Thin Client
Display
•Financial
•Asset
•Planning
•Procuremen
•Distributio
•Human
Technology
•Java
•Java RMI
•Java RMI
•Web Server
•Jini
•Windows NT
Embedded, JVM
•Windows CE, JVM
•Embedded OS,
66
56
58
68
62
60
64
Field Device
70
Control Network
TCP/IP, 10/100 MB Ethernet
One of many
supported
Fieldbusses

Fig. 2

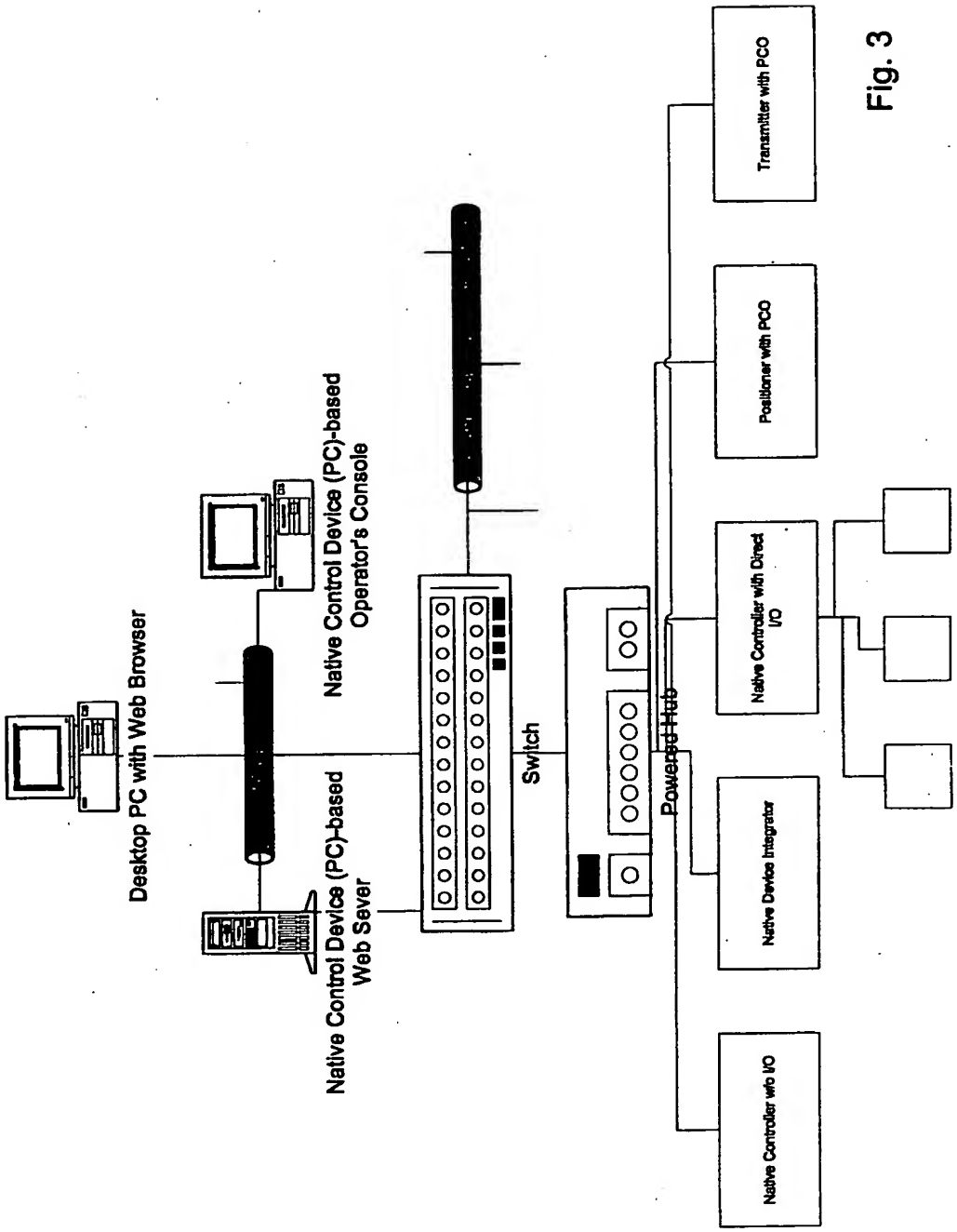

Desktop PC with Web Browser
Native Control Device (PC)-based Operator's Console
Native Control Device (PC)-based Web Sever
Switch
Powered Hub
Transmitter with PCO
Positioner with PCO
Native Controller with Direct I/O
Native Device Integrator
Native Controller w/o I/O

Fig. 3

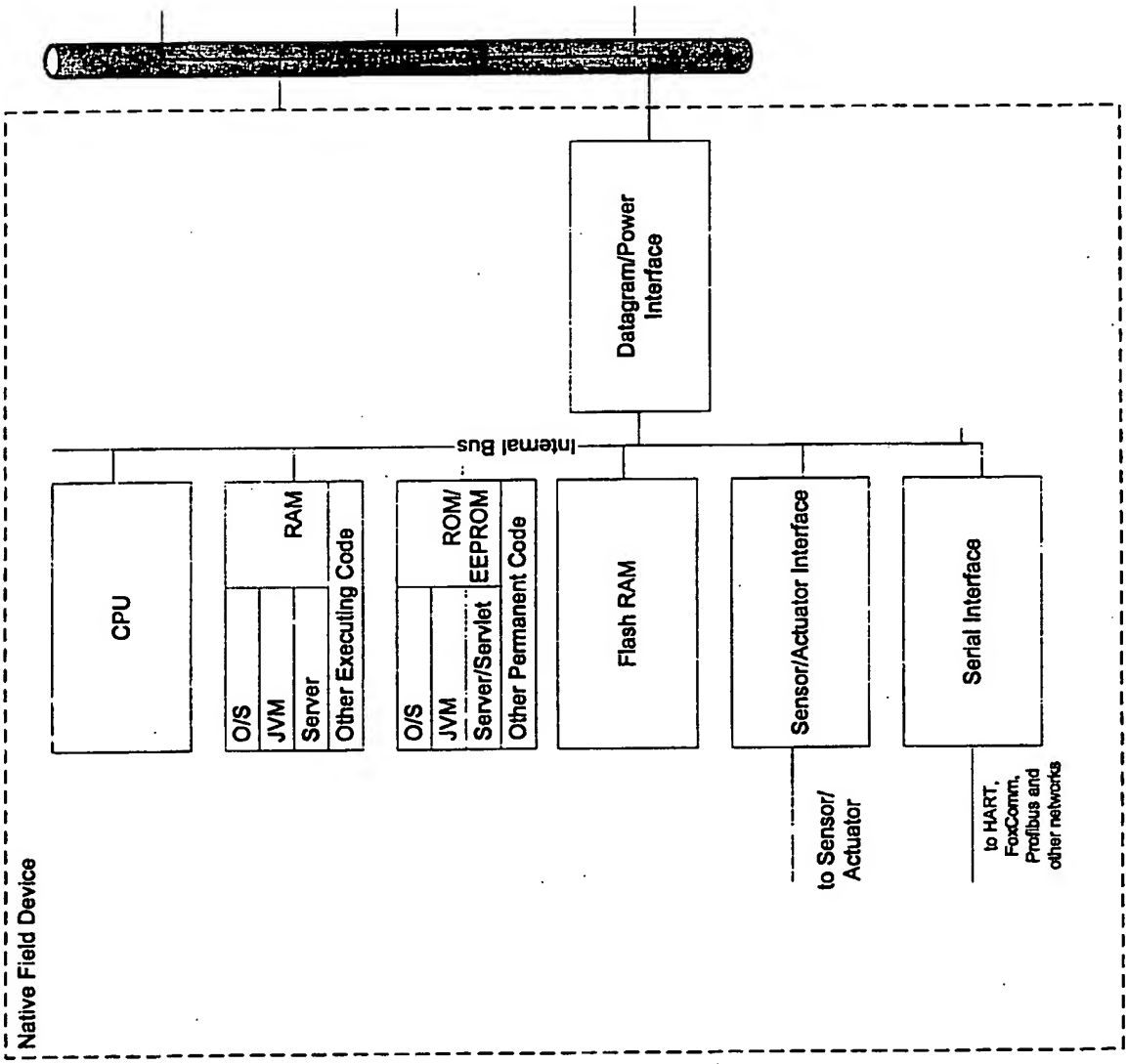

Native Field Device
Datagram/Power Interface
Internal Bus
CPU
RAM
O/S
JVM
Server
Other Executing Code
ROM/ EEPROM
O/S
JVM
Server/Servlet
Other Permanent Code
Flash RAM
Sensor/Actuator Interface
Serial Interface
to Sensor/ Actuator
to HART, FoxComm, Profibus and other networks

Fig. 4

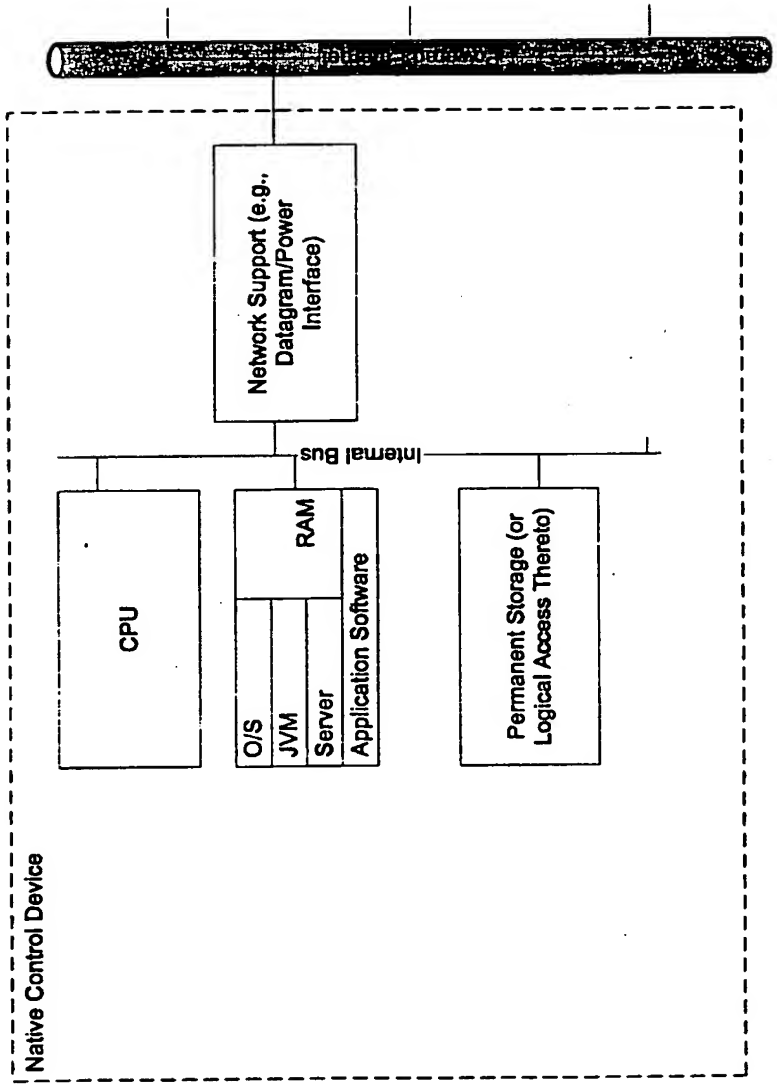

Native Control Device
Network Support (e.g., Datagram/Power Interface)
Internal Bus
CPU
RAM
O/S
JVM
Server
Application Software
Permanent Storage (or Logical Access Thereto)

Fig. 5

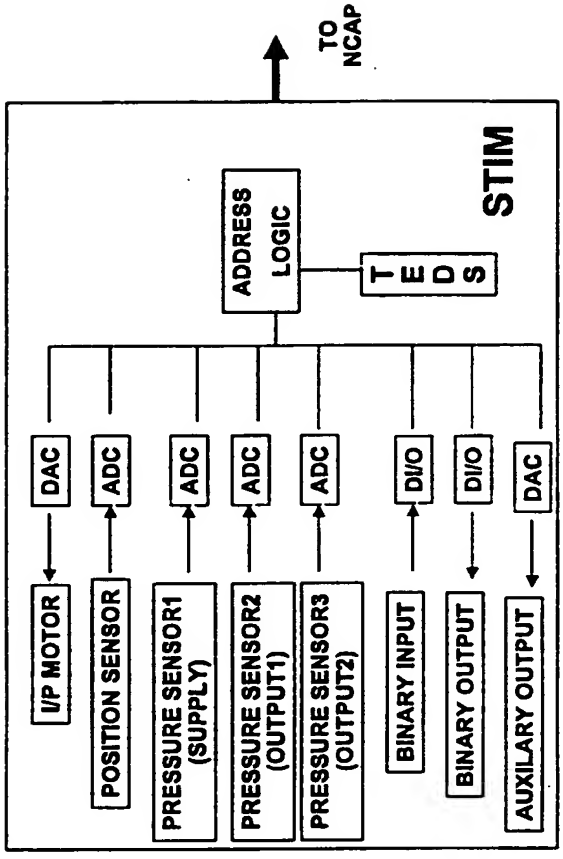

TO NCAP
STIM
ADDRESS LOGIC
TEDS
DAC
ADC
ADC
ADC
ADC
DI/O
DI/O
DAC
I/P MOTOR
POSITION SENSOR
PRESSURE SENSOR1 (SUPPLY)
PRESSURE SENSOR2 (OUTPUT1)
PRESSURE SENSOR3 (OUTPUT2)
BINARY INPUT
BINARY OUTPUT
AUXILARY OUTPUT

Fig. 6